Universidad de las Ciencias Informáticas

Facultad 7

Título: Implementación del Módulo Vectores del Sistema Control Sanitario Internacional

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autores:
Malena Guzmán Pérez
Daybert Hernández Hernández

Tutora: Ing. Yunaysy Ortíz Batista

Ciudad de La Habana
Julio de 2008
"Año 50 de la Revolución"

"Sólo es útil el conocimiento que nos hace mejores."

Sócrates

"El éxito de los hombres no se mide por su éxito inmediato, sino por su éxito definitivo: no se mide por el dinero que acumularon, sino por el resultado de sus obras."

José Martí

Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los 2 días del mes de Julio del año 2008.

_____ _____

Malena Guzmán Pérez *Daybert Hernández Hernández*

Autora **Autor**

_____

Ing. Yunaysy Ortiz Batista

Tutora

Datos de Contacto

Tutora:

Ing. Yunaysy Ortiz Batista

Universidad de las Ciencias Informáticas, La Habana, Cuba.

E-mail: yunaysy@uci.cu

Agradecimientos

A los *que de una forma u otra ayudaron a la realización de este trabajo (Danielito, Ricardo, Dunior, Yanosky, Enrique, Jorge Carlos, Oslien), mil gracias les doy.*
A todos *mis amigos.*
A mi tutora Yunaysy*, al tribunal de tesis y a mi oponente Fals: gracias por la comprensión y el apoyo.*
Lester: *Mi amigo y maestro. Gracias por todo lo que has hecho por mí. Ojalá en el futuro todo lo pueda resolver de forma tan sencilla como cuando me decías: "chama, la pincha es esta....."*
Yeilyn: *De más está decirte lo mucho que te quiero, aunque se que al leer esto me dirás: "yo más a ti". Gracias por tu amistad y tu sinceridad desmedida pero oportuna.*
May: *No me alcanzaría la hoja de agradecimientos para ponerte todo lo que te debo y te agradezco. Me gustaría ser el mejor amigo del mundo para poder convertirme en la mitad del amigo que mereces.*
A toda mi familia *por el pedacito de educación que han puesto en mí. Aunque crean que no, todos me han ayudado a ser lo que soy hoy.*
Male: *Sin ti, nada de estoy hubiese sido posible, eres una de las mejores cosas que me ha pasado en la vida. Gracias por quererme y soportarme.*
MamiChina: *Gracias por ser mi mamá postiza y quererme como si fuera tu hijo de verdad.*
Hiroshy: *Llegaré a sentirme un verdadero profesional en mi campo el día que logre ser como tú. Eres mi ejemplo a seguir, y lo creas o no, para mí, eres "El Mejor". Gracias por toda tu ayuda y apoyo.*
Shairo, Tata, Papá, Mamá: *Ustedes son mi vida, no se qué ponerles… No sé qué agradecerles porque al decirles "todo" me faltan cosas por mencionar. No sé qué decirles que pueda darles una medida de cuan importantes y necesarios son para mí. Son la mejor familia que una persona puede tener, y como siempre todo lo nuestro ha sido de todos esto es de ustedes y para ustedes también.*

Daybert

A todos *los que de una forma u otra contribuyeron a la realización de este trabajo (Dunior, Ricky, Danielito, Sergui, Jorgito, Yano, Runer).*
A mi tutora Yunaysy*, a mi oponente Fals y al tribunal por comprendernos.*
A mis amigas Yadi, Yau y Niurki *por estar a mi lado tanto en los buenos momentos como en los malos.*
A mis compañeros *y* ***profesores*** *de universidad.*
A toda mi familia *por contribuir a que este sueño se hiciera realidad.*
A Julio *por ayudarme con todo lo que me hizo falta sin importar lo majadera que pudiera ser.*
A ti mi vida: ***Daybert****, porque sin ti no hubiese podido seguir adelante, porque fuiste capaz de darme fuerzas cuando también te encontrabas desanimado. Gracias por quererme y soportarme tú a mí también.*
Y por supuesto, ***a mis padres,*** *por apoyarme en todo momento, por ser los mejores del mundo. Quisiera decirles tantas cosas y no encuentro las palabras para expresar todo lo que siento; pero quiero que sepan que todo lo que soy se los debo a ustedes……Gracias por todo.*

Male

Dedicatoria

A todos los profesores que he tenido desde el preescolar hasta hoy.
A todos mis amigos: Johnny, Asmel, Dayan, Alejandro, Yoandy, Lester, Dunior, Serguey, Oslien, El Russo, Josué, Jorge Carlos y Runer.
A mis amigas: May y Yei.
A todos mis compañeros de aula y a la gente de las convocatorias (los de la pegada).
A mi familia: Mis abuelos, mis tíos, mis primos y a MamiChina.
A mi Malena.
A mi tata Lidyse, mi sobrino Shairo y a mi cuñado Hiroshy.
A mis padres Evelyn y Julio; esta tesis es para ustedes más que para nadie, espero de algún modo poder hacerlos sentir orgullosos de mi.

Daybert

A mi mamita y mi papito por ser los mejores padres que una hija pudiera desear.
A mi Daybertititico.
A mimi y nana por ser como unas madres para mi.
A mis abuelitos: mimona y pipon.
A toda mi familia en general.
A mis amigas: Yadi, Yau y Niurki.
Y a todos los profesores que contribuyeron a que me realizara como profesional.

Male

Resumen

Con la realización de este trabajo se pretende llevar a cabo la implementación de un sistema automatizado Web; capaz de gestionar la información referente al trabajo realizado en el programa de Vigilancia y Lucha Antivectorial; específicamente en la batalla por la eliminación del vector mosquito *Aedes Aegypti*.

Surge por la necesidad de contar con una herramienta capaz de efectuar de manera más efectiva y organizada la vigilancia y control de este mosquito, mediante la gestión de la información, que resulta imprescindible para la toma de decisiones.

Para el desarrollo del sistema se utilizaron herramientas y tecnologías predefinidos por el Área Temática Sistemas Especializados para la Salud y el grupo de arquitectura MINSAP-MIC; como el framework CodeIgniter, el lenguaje de programación PHP 5 y el gestor de Base de Datos MySQL 5.

La aplicación cuenta con funcionalidades para llevar un control sistemático de las inspecciones diarias a locales y locales positivos encontrados. Permite gestionar los datos utilizados para realizar estas inspecciones como son: los nomencladores, la configuración de ciclos y de aéreas de salud.

Con la utilización de este sistema informático se espera lograr el flujo de información actualizada, tan importante en la toma de decisiones; sustituir el engorroso trabajo con documentos por la utilización de la aplicación ganando en limpieza y claridad de los datos, y minimizar la introducción de errores humanos.

Palabras claves: Implementación, Control Sanitario Internacional, Vectores, Arquitectura, Base de Datos, CodeIgniter.

Tabla de Contenidos

Tabla de Contenidos

Introducción

En los últimos tiempos, la aplicación de las Tecnologías de la Información y las Comunicaciones (TIC) se ha convertido en parte del sustrato tecnológico del flujo de información en el cual está inmerso el mundo, por lo que se ha hecho imperativa la informatización y automatización de todo proceso cuya carga de procesamiento sea alta y perfectamente sustituible por nuevas tecnologías.

Debido a esto, Cuba enfrenta hoy el enorme reto de llevar a cabo la informatización de la sociedad y la optimización del uso de las Nuevas Tecnologías de la Información y las Comunicaciones.

Organismos como el Ministerio de Salud Pública (MINSAP) necesitan prioridad, debido a la importante tarea que sectores como el de la Salud llevan a cabo en busca del perfeccionamiento de los servicios que se brindan a la población, es por esto, que dentro de este propio sector este reto está tomando lugar.

Los primeros pasos en la utilización de las técnicas de computación para la salud en Cuba comienzan en los primeros años del triunfo de la Revolución, sobre todo aplicada a la gestión administrativa. A finales de la década del 60 y principios del año 1970, se amplía su campo de acción. Su uso comienza a generalizarse con fines médicos específicos, pues se sintió la necesidad de organizar la información en forma sistemática y minuciosa, debido a la gran cantidad de datos a computar y la complejidad de interrelacionar los mismos con el fin de acumularlos para su ulterior procesamiento.

Actualmente, el uso de la informática en la medicina ha permitido al sector de la salud contar con técnicas novedosas y eficaces, no solo para la gestión de información administrativa, sino también para mejorar la organización, control y estadísticas en este sector; así como para optimizar la simulación de modelos concretos de investigación, diagnóstico automatizado y diferencial, pronóstico y selección de tratamientos.

Precisamente, dentro del sector de la salud se encuentra el programa de Control Sanitario Internacional, el cual ha cobrado gran importancia debido a que el tráfico internacional se ha desarrollado de manera intensa en estos últimos tiempos como resultado del aumento de las relaciones internacionales y el surgimiento de programas médicos y de formación masiva de personal, lo que ha determinado un creciente intercambio de viajeros, fundamentalmente con países subdesarrollados, que en los últimos años ha significado un incremento del riesgo de entrada de

personas enfermas, portadores, hospederos intermediarios o vectores, condicionando así la posibilidad de transmisión de enfermedades desconocidas o ya erradicadas en Cuba.

Este programa está conformado por tres subprogramas: el programa de Higiene y Epidemiología que se encarga de la vigilancia epidemiológica al viajero, el programa de Salud Ambiental que estudia los factores del ambiente y del entorno que afectan la salud de humanos, animales y vegetales; y el programa de Vigilancia y Lucha Antivectorial que tiene la misión de contribuir a evitar la introducción y/o propagación de enfermedades introducidas por vectores e incrementar los niveles de salud y satisfacción de la población cubana.

Este último constituye actualmente uno los programas priorizados del MINSAP, por el elevado riesgo de introducción de enfermedades de transmisión vectorial a que constantemente Cuba está sometida, como el hantavirus, la fiebre amarilla, la fiebre del nilo occidental y principalmente el dengue, cuya situación internacional es cada vez más grave con tendencia al empeoramiento cada año.

La vigilancia y control de los vectores efectivos, fundamentalmente, el mosquito *Aedes Aegypti*, se realiza mediante la recogida de información. Esta es utilizada por los especialistas que deben monitorear todos estos eventos relacionados con dengue y otras enfermedades foráneas, con el fin de prevenir epidemias.

La mayor parte de la recogida, procesamiento y transmisión de los datos se realiza manualmente, por vía telefónica o en algunos casos mediante computadoras. En estas, se genera un documento Excel que recoge la información para luego ser distribuida y/o almacenada, lo que hace una realidad la demora de estos procesos a la hora de contar con esta información para llevar a cabo los análisis pertinentes. Provocando que la gestión de la información se vea afectada en rapidez causando grandes problemas a la hora de la toma de decisiones, impidiendo la eficiencia en la actividad de Vigilancia y Lucha Antivectorial.

Tras un profundo análisis de la **Situación Problémica** existente y dada la necesidad de encontrar una solución para dar respuesta a estas dificultades se centrarán los esfuerzos hacia una dirección única, resolver el siguiente problema: ¿Cómo viabilizar la gestión de la información relacionada con el control de focos de mosquitos *Aedes Aegypti* en Cuba?

Introducción

Como **Objeto de Estudio** se ha identificado el proceso de gestión de la información referente al Control Sanitario Internacional en Cuba, teniendo en cuenta que la investigación se centra en el proceso de gestión de la información referente al control de focos de mosquitos *Aedes Aegypti*, siendo este el **Campo de Acción**.

Se traza como **Objetivo General**: Implementar una aplicación Web que permita viabilizar la gestión de la información relacionada con el control de focos de mosquitos *Aedes Aegypti* en Cuba.

Para alcanzar el objetivo mencionado anteriormente se plantean las siguientes tareas de investigación:

✦ Realizar un análisis de las tecnologías, lenguaje y herramientas a utilizar para el desarrollo de la aplicación, así como de otras tecnologías, herramientas y lenguajes utilizados para el desarrollo de aplicaciones web.
✦ Seleccionar estilos o patrones arquitectónicos.
✦ Estructurar el modelo de implementación.
✦ Realizar el diseño de la base de datos.
✦ Realizar la implementación del módulo.
✦ Integrar la aplicación con componentes existentes en el Sistema de Información para la Salud (SISalud).

Este documento se encuentra compuesto por tres capítulos donde se refleja todo el trabajo investigativo, así como todo lo referente al diseño de la base de datos y la implementación de la solución propuesta, distribuido de la siguiente manera:

Capítulo 1. Fundamentación teórica
Se realiza un análisis de las tendencias actuales, lenguajes, plataformas, librerías y herramientas más utilizadas en el desarrollo de aplicaciones web, concluyendo el mismo con una explicación de las tecnologías y herramientas que fueron utilizadas para llevar a cabo la solución del problema que se enfrenta.

Capítulo 2. Elementos de arquitectura
Se realiza una explicación de la arquitectura del sistema, para lo que se exponen los requisitos no funcionales del mismo, se realiza un análisis de los patrones o estilos arquitectónicos presentes en la

propuesta de solución, así como una explicación de la vista de despliegue y la vista de implementación.

Capítulo 3. Descripción y análisis de la solución propuesta

Se realiza un análisis de la estrategia de integración con otros módulos o sistemas, la descripción de de la clases y métodos más importantes, así como el Diagrama Entidad Relación (DER) de la Base de Datos (BD) y por último una descripción de las tablas de la misma.

Capítulo 1: Fundamentación Teórica

Introducción

Este capítulo tiene como objetivo fundamental abordar distintos aspectos que se utilizan como soporte teórico para el desarrollo de la aplicación. Se realiza un análisis de las tendencias actuales, lenguajes, y herramientas más utilizadas en el desarrollo de aplicaciones web, y se presentan las tecnologías y herramientas a utilizar para llevar a cabo la solución del problema que se enfrenta.

1.1 Tendencias y tecnologías actuales

1.1.1 Internet

Se puede definir como todos le conocen “La Red de Redes” o “La Autopista de la Información”, es decir, una red que no sólo interconecta computadoras, sino que interconecta redes de computadoras entre sí; una red de computadoras es un conjunto de máquinas que se comunican a través de algún medio (cable coaxial, fibra óptica, radiofrecuencia y líneas telefónicas) con el objeto de compartir recursos; de esta manera, Internet sirve de enlace entre redes más pequeñas y permite ampliar su cobertura al hacerlas parte de una "red global".

Internet no es más que un método de interconexión descentralizada de redes de computadoras implementado en un conjunto de protocolos denominado TCP/IP y garantiza que redes físicas heterogéneas funcionen como una red lógica única, de alcance mundial. Sus orígenes se remontan a 1969, cuando se estableció la primera conexión de computadoras, conocida como ARPANET, entre tres universidades en California y una en Utah, EE. UU.

Funciona con la estrategia Cliente/Servidor, un paradigma de división del trabajo informático en el que las tareas se reparten entre un número de clientes que efectúan peticiones de servicios de acuerdo con un protocolo, y un número de servidores que las atienden.

1.1.2 Aplicaciones Web

Las aplicaciones Web son una especialización y concreción de las aplicaciones Cliente/Servidor. Una aplicación web es una aplicación informática que permite a los usuarios acceder a un servidor a través de la red. La popularidad de las aplicaciones web se debe principalmente a los navegadores como clientes ligeros y que para actualizar y mantener las aplicaciones no es necesario distribuir e instalar software en miles de potenciales clientes. En términos más simples una aplicación web es un Sistema

Web que permite a los usuarios ejecutar la lógica de negocio a través de un navegador. Las aplicaciones web hacen uso de las tecnologías que existen para permitir a los usuarios del sistema modificar la lógica del negocio en el servidor y para generar contenidos dinámicos; de no existir lógica de negocio en el servidor el sistema es considerado como sitio no como aplicación web.

Las aplicaciones web generalmente presentan una arquitectura simple, como componentes principales usan el servidor web, la red y el navegador. El servidor es el encargado de distribuir las páginas por cada petición de los clientes. Las peticiones se hacen a través de las conexiones de red y para hacerlo utilizan el protocolo de comunicación HTTP (HyperText Transfer Protocol). Para mostrar la información a los usuarios se usa siempre un browser o navegador.

Principales ventajas de las aplicaciones web: (1)

- Una empresa puede migrar de sistema operativo o cambiar el Hardware libremente sin afectar el funcionamiento de las aplicaciones del servidor.
- No se requieren complicadas combinaciones de Hardware/Software para utilizar estas aplicaciones. Solo un computador con un buen navegador web.
- Se facilita el trabajo a distancia. Se puede trabajar desde cualquier PC o computador portátil con conexión a Internet.
- Actualizar o hacer cambios en el software es sencillo y sin riesgos de incompatibilidades. Existe solo una versión en el servidor lo que implica que no hay que distribuirla entre los demás computadores. El proceso es rápido y limpio.
- La utilización de ésta tecnología conlleva a reducir costos y complicaciones, y proporciona mayor libertad a la hora de realizar cualquier tipo de cambios.

Algunas desventajas: (2)

- Acceso limitado. La necesidad de conexión permanente y rápida a Internet hacen que el acceso a estas aplicaciones no esté al alcance de todos.
- La interactividad no se produce en tiempo real, en las aplicaciones web cada acción del usuario conlleva un tiempo de espera excesivo hasta que se obtiene la reacción del sistema.
- Elementos de interacción muy limitados. En comparación con el software de escritorio, las posibilidades de interacción con el usuario que ofrecen las aplicaciones web (mediante formularios principalmente) son muy escasas.
- Diferencias de presentación entre plataformas y navegadores. La falta de estándares ampliamente soportados dificulta el desarrollo de las aplicaciones.

1.1.3 El Navegador Web o Browser

El cliente para solicitar y visualizar la información que desea necesita un programa que le sirva de interfaz. Estos programas se conocen como browser o navegadores. Pueden considerarse como una interfaz de usuario universal. Dentro de sus funciones están la petición de las páginas Web, la representación adecuada de sus contenidos y la gestión de los posibles errores que se puedan producir.

Las responsabilidades de los browsers se pueden resumir en: permitir que el usuario pida una dirección URL (Uniform Resource Locators) para solicitar un recurso en la red, decodificar la URL, conectarse a una computadora servidora, recibir la página de resultado que envía el host, interpretar el hipertexto encontrado, mostrarlo adecuadamente a las características y limitaciones del entorno del cliente, recoger el resto de los elementos que forman la página web como imágenes, sonidos y objetos insertados. Además debe responder a las acciones que ejecuta el usuario como activación de un hipervínculo y otros eventos accionados en el navegador que pueden requerir el desencadenamiento de una acción en el cliente.

Para todo esto, los fabricantes de navegadores les han dotado de posibilidades de ejecución de programas de tipo script, con modelos de objetos que permiten manipular los contenidos de los documentos. Estos lenguajes de programación son VBScript, JScript (ambas de Microsoft) y JavaScript (de Netscape), y proporcionan las soluciones llamadas del lado del cliente, client side, y permiten realizar validaciones de datos recogidos en las páginas antes de enviarlos al servidor y proporcionan un alto grado de interacción con el usuario dentro del documento. (3)

Otras de las posibilidades de los navegadores es la gestión del llamado HTML dinámico (Dinamic HTML, DHTML). Éste está compuesto de HTML, hojas de estilo en cascada, (Cascade Style Sheets, CSS), modelo de objetos y scripts de programación que permiten formatear y posicionar correctamente los distintos elementos HTML de las páginas Web, permitiendo un mayor control sobre la visualización de las páginas. (4)

En esta línea, los navegadores han ido un poco más allá y permiten las visualizaciones de documentos XML (eXtensible Markup Language) después de haber sido transformados adecuadamente a HTML por las hojas de estilo extensibles (eXtensible Style Sheets, XSL). De esta manera se puede elegir visualizar ciertos elementos y otros no, dependiendo de las circunstancias. (5)

1.1.4 HyperText Transfer Protocol (HTTP)

Cada transacción de información realizada en la Web es realizada utilizando el protocolo HTTP, "HyperText Transfer Protocol" por sus siglas en inglés, o Protocolo de Transferencia de HyperTexto.

De este modo, las peticiones de acceso a una página y la respuesta brindada por la misma en forma de contenido de hipertexto utilizan este sistema de comunicación, el cual permanece un tanto oculto al usuario final. El protocolo HTTP es utilizado también para enviar formularios con campos de texto, u otro tipo de información en ambos sentidos. (6)

La conexiones realizadas mediante este protocolo no son guardadas por el mismo en ningún sitio, es decir que se trata de un protocolo "sin estado": los datos se pierden, por lo tanto, cuando la transacción de los mismos ha terminado, cosa que da lugar a las cookies, archivos livianos que se guardan en determinado sitio del disco duro con el objetivo de almacenar información del usuario. De tal forma, el sitio Web sabrá de quién se trata al volver a acceder al mismo, mostrando por ejemplo su nombre y/ o permitiendo su acceso sin necesidad de ingresar contraseña. (7)

Entre las propiedades de HTTP se pueden destacar las siguientes: (8)

- Un esquema de direccionamiento comprensible.
  Utiliza el Universal Resource Identifier (URI) para localizar sitios URL o nombres (URN, Uniform Resource Name) sobre los que hay que aplicar un método.
- Arquitectura Cliente/Servidor.
  HTTP se asienta en el paradigma solicitud/respuesta. La comunicación se asienta sobre TCP/IP. El puerto por defecto es el 80.
- Es un protocolo sin conexión y sin estado.
  Después de que el servidor ha respondido la petición del cliente, se rompe la conexión entre ambos. Además no se guarda memoria del contexto de la conexión para siguientes conexiones.
- Está abierto a nuevos tipos de datos.
  HTTP utiliza tipos MIME (Multipart Internet Mail Extension) para la determinación del tipo de los datos que transporta. Cuando un servidor HTTP transmite información de vuelta a un cliente, incluye una cabecera que le indica al cliente sobre los tipos de datos que componen el documento. De la gestión de esos datos se encargan las utilidades que tenga el cliente (visor de imágenes y de vídeo).

1.2 Servidor Web

El desarrollo de las tecnologías y la informática permitió que apareciera Internet como la red de redes, la red de computadoras más grande que existe y en la que toda su información se encuentra pública.

Así, como medio para presentar y acceder a la información se ha creado un importante sector, el World Wide Web o web y en el cual se presenta la información por medios de páginas web y se acceden a través del modelo Cliente/Servidor.

Para que esto sea posible debe existir el servidor web, el cual no es más que un programa que sirve para atender y responder a las diferentes peticiones de los navegadores, proporcionando los recursos que soliciten usando el protocolo HTTP.

Un servidor web básico cuenta con un esquema de funcionamiento muy simple, basado en ejecutar infinitamente el siguiente bucle:

- Espera peticiones en el puerto TCP indicado (el estándar por defecto para HTTP es el 80).
- Recibe una petición.
- Busca el recurso.
- Envía el recurso utilizando la misma conexión por la que recibió la petición.
- Y vuelve al segundo punto.

Es decir, el servidor web escucha las peticiones HTTP que le llegan y las satisface. Dependiendo del tipo de la petición, buscará una página Web o bien ejecutará un programa en el servidor. De cualquier modo, siempre devolverá algún tipo de resultado HTML al cliente o navegador que realizó la petición. (9)

Para ello en las computadoras servidoras se almacenan un conjunto de páginas web como sitio web así como otro tipo de archivos necesarios para presentar las páginas web como imágenes, sonidos y bases de datos, lo que permite dada una solicitud de información de un cliente hacer un procesamiento interno para presentarle la información de la forma más apropiada.

Los servidores web permiten alojar sitios web y brindar esta información de forma pública o restringida a los clientes. Estas computadoras deben de contar con altos requerimientos de velocidad, memoria y

espacio en disco duro para ser capaces de soportar todas las peticiones que se les hace simultáneamente y no colapsar por necesidad de recursos.

Básicamente, un servidor web sirve contenido estático a un navegador, carga un archivo y lo sirve a través de la red al navegador de un usuario. Este intercambio es mediado por el navegador y el servidor que hablan el uno con el otro mediante HTTP. (10) Funcionan las 24 horas del día, 365 días al año.

1.2.1 Servidor Web Apache

Apache es el servidor web hecho por excelencia, su configurabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. (11)

La licencia Apache es una descendiente de la licencias BSD (Berkeley Software Distribution), no es GPL (General Public License). Esta licencia permite hacer lo que se desee con el código fuente (incluso productos propietarios). (12)

Algunas razones por las que este software libre es grandemente reconocido en muchos ámbitos empresariales y tecnológicos son: (13)

- Corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal.
- Apache es una tecnología gratuita de código fuente abierto. El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto. Esto le da una transparencia a este software de manera que si se quiere ver que es lo que se está instalando como servidor se puede saber, sin ningún secreto, sin ninguna puerta trasera.
- Es un servidor altamente configurable de diseño modular. Actualmente existen muchos módulos para Apache que son adaptables a este, y están ahí para que sean instalados cuando sea necesario.
- Trabaja con gran cantidad de Perl, PHP y otros lenguajes de script. También trabaja con Java y páginas jsp. Teniendo todo el soporte que se necesita para tener páginas dinámicas.
- Apache permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.

- Tiene una alta configurabilidad en la creación y gestión de logs. Permite la creación de ficheros de log a medida del administrador y de este modo se puede tener un mayor control sobre lo que sucede en el servidor.

1.2.2 Internet Information Server (IIS)

Internet Information Services (IIS) es una serie de servicios para los ordenadores que funcionan con Windows. Originalmente era parte del Option Pack para Windows NT. Luego fue integrado en otros sistemas operativos de Microsoft destinados a ofrecer servicios, como Windows 2000 o Windows Server 2003. Windows XP Profesional incluye una versión limitada de IIS. Los servicios que ofrece son: FTP, SMTP, NNTP y HTTP/HTTPS.

Este servicio convierte a un ordenador en un servidor de Internet o Intranet es decir que en las computadoras que tienen este servicio instalado se pueden publicar páginas web tanto local como remotamente (servidor web).

El servidor web se basa en varios módulos que le dan capacidad para procesar distintos tipos de páginas, por ejemplo Microsoft incluye los de Active Server Pages (ASP) y ASP.NET. También pueden ser incluidos los de otros fabricantes, como PHP o Perl.

1.3 Lenguajes de programación Web

La programación Web, parte de las siglas WWW, que significan World Wide Web o telaraña mundial. Esta incluye tres conceptos fundamentales que se deben conocer antes de hacer una página con cualquier lenguaje de programación web. Uno de ellos es el URL, sistema con el cual se localiza un recurso dentro de la red, este recurso puede ser una página web, un servicio o cualquier otra cosa; en resumen el URL no es más que un nombre, que identifica una computadora.

El siguiente concepto dentro de la programación Web, es el protocolo encargado de llevar la información que contiene una página Web por toda la red de internet, como es el HTTP; y por último el lenguaje necesario cuya funcionalidad es la de representar cualquier clase de información que se encuentre almacenada en una página Web, este lenguaje es el HTML (Hypertext Markup Language). En la programación Web, el HTML es el lenguaje que permite codificar o preparar documentos de hipertexto, que viene a ser el lenguaje común para la construcción de una página Web.

1.3.1 Lenguajes del lado del cliente

La parte del cliente de las aplicaciones web está formada por el código HTML que forma la página web, con opción a código ejecutable mediante los lenguajes de scripting de los navegadores (JavaScript) o mediante pequeños programas (applets) en Java.

Los lenguajes scripts son lenguajes interpretados, no compilados; lo que significa que el programa que lo interpreta (el navegador en este caso) va leyendo el código y ejecutándolo al mismo tiempo en vez de transformarlo a código máquina, lo que puede traer errores en tiempo de ejecución. También es normal que los lenguajes scripts no posean un control estricto de tipos de datos para simplificar la programación.

La principal ventaja de un lenguaje interpretado es que es independiente de la máquina y del sistema operativo ya que no contiene instrucciones propias de un procesador sino que contiene llamadas a funciones que el intérprete deberá reconocer. Basta que exista un intérprete de un lenguaje para dicho sistema y todos los programas escritos en ese lenguaje podrán ejecutarse en ese sistema.

Además un lenguaje interpretado permite modificar en tiempo de ejecución el código que se está ejecutando, así como añadirle nuevo, algo que resulta práctico cuando se quiere hacer pequeñas modificaciones en una aplicación y no se desea tener que recompilarla toda cada vez.

O sea, los lenguajes de lado cliente (entre los cuales no sólo se encuentra el HTML sino también el Java y el JavaScript los cuales son simplemente incluidos en el código HTML) son aquellos que pueden ser directamente interpretados por el navegador y no necesitan un pre-tratamiento.

1.3.1.1 HyperText Markup Language (HTML)

HTML es un lenguaje muy sencillo que permite describir hipertexto, es decir, texto presentado de forma estructurada y agradable, con enlaces que conducen a otros documentos o fuentes de información relacionadas, y con inserciones multimedia; es el lenguaje que se utiliza para presentar información en el World Wide Web.

La descripción se basa en especificar en el texto la estructura lógica del contenido (títulos, párrafos de texto normal, enumeraciones, definiciones y citas), así como los diferentes efectos que se quieren dar

(cursiva, negrita, o un gráfico determinado) y dejar que luego la presentación final de dicho hipertexto se realice por un programa especializado (como Internet Explorer o Firefox).

HTML es un lenguaje de marca, o sea, una manera de expresar la información de un documento (por ejemplo: información sobre los vínculos del hipertexto y sobre formato) en el documento mismo. Los lenguajes de marca, usan etiquetas que son marcas que se ubican dentro del texto y que brindan información de despliegue. De este modo, el Lenguaje Marca de Hipertexto es una forma específica de usar etiquetas para ofrecer información sobre un documento.

Este lenguaje indica al navegador donde colocar cada texto, cada imagen o cada video y la forma que tendrán estos al ser colocados en la página. Así que no es más que una serie de etiquetas que se utilizan para definir la forma o estilo que se quiera aplicar al documento.

1.3.1.2 JavaScript

Se trata de un lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento. Gracias a su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación del lado del cliente más utilizado.

Es un lenguaje de programación interpretado por el navegador que se utiliza para controlar su apariencia y manipular los eventos que ocurran en su ventana. A través del JavaScript se pueden conseguir interesantes efectos en las páginas web, validar formularios, abrir y cerrar ventanas, cambiar dinámicamente el aspecto y los contenidos de una página, realizar cálculos matemáticos sencillos.

Con este se puede crear efectos especiales en las páginas y definir interactividades con el usuario. El navegador del cliente es el encargado de interpretar las instrucciones JavaScript y ejecutarlas para realizar estos efectos e interactividades, de modo que el mayor recurso, y tal vez el único, con que cuenta este lenguaje es el propio navegador. (14)

Es un lenguaje de programación bastante sencillo y pensado para hacer las cosas con rapidez, a veces con ligereza. Incluso las personas que no tengan una experiencia previa en la programación podrán aprender este lenguaje con facilidad y utilizarlo en toda su potencia con sólo un poco de práctica.

Entre las acciones típicas que se pueden realizar con este lenguaje se tienen dos vertientes. Por un lado los efectos especiales sobre páginas web, para crear contenidos dinámicos y elementos de la página que tengan movimiento, cambien de color o cualquier otro dinamismo. Por el otro, JavaScript permite ejecutar instrucciones como respuesta a las acciones del usuario, con lo que se puede crear páginas interactivas con programas como calculadoras, agendas, o tablas de cálculo. (15)

Es un lenguaje con muchas posibilidades, permite la programación de pequeños scripts, pero también de programas más grandes, orientados a objetos, con funciones, estructuras de datos complejas. Además, pone a disposición del programador todos los elementos que forman la página web, para que éste pueda acceder a ellos y modificarlos dinámicamente. (16)

Con JavaScript el programador se convierte en el verdadero dueño y controlador de cada cosa que ocurre en la página cuando la está visualizando el cliente.

1.3.1.3 Visual Basic Script (VbScript)

Es un lenguaje de programación de scripts del lado del cliente basado en Visual Basic, un popular lenguaje para crear aplicaciones Windows. Tanto su sintaxis como la manera de trabajar están muy inspiradas en él. Sin embargo, no todo lo que se puede hacer en Visual Basic se podrá hacer en Visual Basic Script, pues este último es una versión reducida del primero.

El modo de funcionamiento de Visual Basic Script para construir efectos especiales en páginas web es muy similar al utilizado en JavaScript y los recursos a los que se puede acceder también son los mismos: el navegador. (17)

El crecimiento del uso de las tecnologías de internet ha supuesto un significativo avance para este lenguaje, dado que es parte fundamental de la ejecución de aplicaciones de servidor programadas en ASP pero los desarrolladores de aplicaciones web suelen preferir JavaScript debido a su mayor compatibilidad con otros navegadores de Internet, ya que VBScript sólo está disponible para el navegador de Microsoft Internet Explorer, y no en otros como Firefox u Opera.

1.3.2 Lenguajes del lado del servidor

La programación del lado del servidor es un elemento agregado muy importante en el diseño o construcción de sitios Web, ya que permite de una u otra forma el manejo de datos dinámicamente.

Esta permite disponer de un entorno de aplicación mucho más complejo del que se podría conseguir solamente con HTML y la programación en el lado del cliente. Por ejemplo, se podría necesitar acceder a algunos datos, o incluso programas de un gran sistema. Esos programas podrían trabajar sobre una cantidad considerable de datos y enviar los resultados. Pondríamos entonces poner los datos resultantes en una base de datos para una posterior manipulación, antes de generar la página final que se enviará al cliente.

Así pues, se puede hablar de lenguajes de lado servidor que son aquellos lenguajes que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él.

1.3.2.1 Hypertext Preprocessor (PHP)

PHP (acrónimo de Hypertext Preprocessor) es un lenguaje ”del lado del servidor” (esto significa que PHP funciona en un servidor remoto que procesa la página Web antes de que sea abierta por el navegador del usuario) especialmente creado para el desarrollo de páginas Web dinámicas. Puede ser incluido con facilidad dentro del código HTML, y permite una serie de funcionalidades tan extraordinarias que se ha convertido en el favorito de millones de programadores en todo el mundo. Es un lenguaje gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación.

Es conocido como una tecnología de código abierto que resulta muy útil para diseñar de forma rápida y eficaz aplicaciones Web dirigidas a bases de datos, es un potente lenguaje de secuencia de comandos diseñado específicamente para permitir a los programadores crear aplicaciones en Web con distintas prestaciones de forma rápida y puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno.
Combinado con la base de datos MySQL, es el lenguaje estándar a la hora de crear sitios de comercio electrónico o páginas Web dinámicas.

Entre sus características fundamentales están:

- Gratuito.
  Al tratarse de software libre, puede descargarse y utilizarse en cualquier aplicación, personal o profesional, de manera completamente libre.
- Gran popularidad.

Existe una gran comunidad de desarrolladores y programadores que continuamente implementan mejoras en su código. Casi 10 millones de páginas Web desarrolladas con PHP.

- Enorme eficiencia.
  Con escaso mantenimiento y un servidor gratuito (Apache), puede soportar muchas visitas diarias.
- Sencilla integración con múltiples bases de datos
  Esencial para una página Web verdaderamente dinámica, es una correcta integración con base de datos. Aunque MySQL es la base de datos que mejor trabaja con PHP, puede conectarse también a PostgreSQL, Oracle, dbm, filepro, interbasem o cualquier otra base de datos compatible con ODBC (Open Database Connectivity Standard).
- Versatilidad.
  Puede usarse con la mayoría de sistemas operativos, ya sea basados en UNIX (Linux, Solares, FreeBSD), como con Windows, el sistema operativo de Microsoft.
- Gran número de funciones predefinidas.
  A diferencia de otros lenguajes de programación, fue diseñado especialmente para el desarrollo de páginas Web dinámicas. Por ello, esta dotado de un gran número de funciones que simplificarán enormemente tareas habituales como descargar documentos, enviar correos, trabajar con cookies y sesiones.

Principales ventajas de PHP:

- Es un lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, se destaca su conectividad con MySQL.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos.
- Posee una amplia documentación en su página oficial entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de Programación Orientada a Objetos.
- Biblioteca nativa de funciones sumamente amplia e incluida.
- No requiere definición de tipos de variables.
- Tiene manejo de excepciones.

Algunas desventajas de PHP:

- No posee una abstracción de base de datos estándar, sino bibliotecas especializadas para cada motor (a veces más de una para el mismo motor).

- No posee adecuado manejo de internacionalización, unicode.
- Por su diseño dinámico no puede ser compilado y es muy difícil de optimizar.
- Por sus características promueve la creación de código desordenado y complejo de mantener.
- Está diseñado especialmente para un modo de hacer aplicaciones web que es ampliamente considerado problemático y obsoleto (mezclar el código con la creación de la página web).

1.3.2.2 Active Server Page (ASP)

ASP (Active Server Pages - Página Activa en el Servidor) no es en sí mismo un lenguaje de programación, si no más bien un marco sobre el que se construyen aplicaciones basadas en Internet, apoyándose para ello en el lenguaje HTML, en lenguajes de script conocidos (generalmente VBScript, pero también JavaScript), en motores de bases de datos y en el lenguaje de consulta SQL. Es una tecnología del lado servidor desarrollada por Microsoft para páginas Web generadas dinámicamente.

Ha pasado por cuatro iteraciones mayores, ASP 1.0 (distribuido con IIS 3.0), ASP 2.0 (distribuido con IIS 4.0), ASP 3.0 (distribuido con IIS 5.0) y ASP.NET (parte de la plataforma .NET de Microsoft). Las versiones pre-.NET se denominan actualmente (desde 2002) como ASP clásico.

ASP no necesita ser compilado para ejecutarse. Existen varios lenguajes que se pueden utilizar para crear páginas ASP. El más utilizado es VBScript, nativo de Microsoft. ASP se puede hacer también en JScript (no JavaScript). El código ASP puede ser insertado junto con el código HTML. Los archivos cuentan con la extensión (asp). (18)

Principales ventajas de ASP: (19)

- Usa Visual Basic Script, siendo fácil para los usuarios.
- Comunicación óptima con SQL Server.
- Soporta el lenguaje JScript (Javascript de Microsoft).

Algunas desventajas de ASP: (20)

- Código desorganizado.
- Se necesita escribir mucho código para realizar funciones sencillas.
- Tecnología propietaria.
- Hospedaje de sitios web costosos.

Desde 2002, el ASP clásico está siendo reemplazado por ASP.NET, que entre otras cosas, reemplaza los lenguajes interpretados como VBScript o JScript por lenguajes compilados a código intermedio (llamado MSIL o Microsoft Intermediate Lenguage) como Visual Basic, C# o cualquier otro lenguaje que soporta la plataforma .NET. Fue desarrollado para resolver las limitantes que brindaba su antecesor. Creado para desarrollar web sencillas o grandes aplicaciones. Los archivos cuentan con la extensión (aspx) y para el funcionamiento de las páginas se necesita tener instalado IIS con el Framework .Net.

Principales ventajas de ASP.NET: (21)

- Completamente orientado a objetos.
- Controles de usuario y personalizados.
- División entre la capa de aplicación o diseño y el código.
- Facilita el mantenimiento de grandes aplicaciones.
- Incremento de velocidad de respuesta del servidor.
- Mayor velocidad.
- Mayor seguridad.

Algunas desventajas de ASP.NET:

- Mayor consumo de recursos.
- Tecnología propietaria.
- Hospedaje de sitios web costosos.

1.3.2.3 Java Server Page (JSP)

Es un lenguaje para la creación de sitios web dinámicos, acrónimo de Java Server Pages. Está orientado a desarrollar páginas web en Java. JSP es un lenguaje multiplataforma. Creado para ejecutarse del lado del servidor. El denominado *contenedor JSP* (que sería un componente del servidor web) es el encargado de tomar la página, sustituir el código Java que contiene por el resultado de su ejecución, y enviarla al cliente.

JSP fue desarrollado por Sun Microsystems. Comparte ventajas similares a las de ASP.NET, desarrollado para la creación de aplicaciones web potentes. Posee un motor de páginas basado en los servlets de Java.

Con JSP se pueden crear aplicaciones web que se ejecuten en variados servidores web, de múltiples plataformas, ya que Java es en esencia un lenguaje multiplataforma. Por tanto, las JSP se pueden escribir con el editor HTML/XML habitual. (22)

La principal ventaja de JSP frente a otros lenguajes es que permite integrarse con clases Java (.class) lo que permite separar en niveles las aplicaciones web, almacenando en clases java las partes que consumen más recursos así como las que requieren más seguridad, y dejando la parte encargada de formatear el documento HTML en el archivo jsp.

Sin embargo JSP no se puede considerar un script al 100% ya que antes de ejecutarse el servidor Web compila el script y genera un servlet, por lo tanto se puede decir que aunque este proceso sea transparente para el programador no deja de ser una aplicación compilada. La ventaja de esto es algo más de rapidez y disponer del API de Java en su totalidad.

Principales características:

- Código separado de la lógica del programa.
- Las páginas son compiladas en la primera petición.
- Permite separar la parte dinámica de la estática en las páginas web.
- Los archivos se encuentran con la extensión (jsp).
- El código JSP puede ser incrustado en código HTML.

Principales Ventajas de JSP:

- Ejecución rápida del servlets.
- Crear páginas del lado del servidor.
- Multiplataforma.
- Código bien estructurado.
- Integridad con los módulos de Java.
- La parte dinámica está escrita en Java.
- Permite la utilización de servlets.

1.4 Sistemas Gestores de Bases de Datos (SGBD)

Un Sistema Gestor o Manejador de Bases de Datos (SGBD) es un conjunto de programas que se encargan de manejar la creación y todos los accesos a las bases de datos, por lo tanto, el SGBD es un

software de propósito general que facilita el proceso de definir, construir y manipular la BD para diversas aplicaciones por lo que el objetivo principal del sistema gestor de base de datos es proporcionar una forma de almacenar y recuperar la información de una base de datos de manera que sea tanto práctica como eficiente.

Existen muchas formas de organizar las bases de datos, pero hay un conjunto de objetivos generales que deben cumplir todas los SGBD, de modo que faciliten el proceso de diseño de aplicaciones y que los tratamientos sean más eficientes y rápidos, dando la mayor flexibilidad posible a los usuarios.

Los objetivos fundamentales de los SGBD son:

- Independencia de los datos y los programas de aplicación.

  Con ficheros ordinarios o tradicionales la lógica de la aplicación contempla la organización de los ficheros y el método de acceso. Entonces es imposible modificar la estructura de almacenamiento o la estrategia de acceso sin afectar el programa de aplicación. Naturalmente, lo que se afecta en el programa son las partes de éste que tratan los ficheros, lo que es ajeno al problema real que el programa de aplicación necesita resolver.

  En un SGBD sería indeseable la existencia de aplicaciones y datos dependientes entre sí, por dos razones fundamentales:

  -Diferentes aplicaciones necesitarán diferentes aspectos de los mismos datos.

  -Se debe poder modificar la estructura de almacenamiento o el método de acceso según los cambios en el fenómeno o proceso de la realidad sin necesidad de modificar los programas de aplicación (también para buscar mayor eficiencia).

  Independencia de los datos: inmunidad de las aplicaciones a los cambios en la estructura de almacenamiento y en la estrategia de acceso y constituye el objetivo fundamental de los SGBD, o sea, es la capacidad de modificar el esquema en un nivel del sistema de BD sin tener que modificar el nivel inmediato superior.

  - Independencia lógica con respecto a los datos: es la capacidad de modificar el esquema conceptual sin tener que alterar los esquemas externos, ni los programas de aplicación. Podemos modificar el esquema conceptual para ampliar la BD o para reducirla. Las modificaciones no deberán afectar los esquemas externos que sólo se refieren a los datos restantes. Además, las restricciones podrán ser modificadas en el esquema conceptual sin afectar los esquemas externos.

- Independencia física respecto a los datos: es la capacidad de modificar el esquema interno sin tener que alterar el esquema conceptual o los externos. Se refiere sólo a la separación entre las aplicaciones y la estructura física de almacenamiento, es más fácil de lograr que la independencia lógica de los datos.

✦ Minimización de la redundancia.

Con los ficheros tradicionales se produce redundancia de la información. Uno de los objetivos de los SGBD es minimizar la redundancia de los datos. Se dice disminuir la redundancia, no eliminarla, pues aunque se definen las BD como no redundantes, en realidad existe redundancia en un grado no significativo para disminuir el tiempo de acceso a los datos o para simplificar el método de direccionado. Lo que se trata de lograr es la eliminación de la redundancia superflua.

✦ Integración y sincronización de las bases de datos.

La integración consiste en garantizar una respuesta a los requerimientos de diferentes aspectos de los mismos datos por diferentes usuarios, de forma que, aunque el sistema almacene la información con cierta estructura y cierto tipo de representación, debe garantizar entregar datos que solicita al programa de aplicación y en la forma en que lo solicita. Está vinculada a la sincronización, que consiste en la necesidad de garantizar el acceso múltiple y simultáneo a la BD, de modo que los datos puedan ser compartidos por diferentes usuarios a la vez. Están relacionadas, ya que lo usual es que diferentes usuarios trabajen con diferentes enfoques y requieran los mismos datos, pero desde diferentes puntos de vista.

✦ Integridad de los datos.

Consiste en garantizar la no contradicción entre los datos almacenados de modo que, en cualquier momento los datos almacenados sean correctos, es decir, que no se detecte inconsistencia entre los datos. Está relacionada con la minimización de redundancia, ya que es más fácil garantizar la integridad si se elimina la redundancia.

✦ Seguridad y protección de los datos.

Protección: garantizar el acceso autorizado a los datos, de forma de interrumpir cualquier intento de acceso no autorizado, ya sea por error del usuario o por mala intención.

Seguridad: que el sistema de bases de datos disponga de métodos que garanticen la restauración de las BD al producirse alguna falla técnica, interrupción de la energía eléctrica.

- Facilidad de manipulación de la información.

- Control centralizado.

  Uno de los objetivos más importantes de los SGBD es garantizar el control centralizado de la información. Permite controlar de manera sistemática y única los datos que se almacenan en la BD, así como el acceso a ella. Lo anterior implica que debe existir una persona o conjunto de personas que tenga la responsabilidad de los datos operacionales: el administrador de la BD, que puede considerarse parte integrante del SGBD.

  Entre las tareas del administrador de la BD están:

  - Decidir el contenido informativo de la BD.
  - Decidir la estructura de almacenamiento y la estrategia de acceso.
  - Garantizar el enlace con los usuarios.
  - Definir los chequeos de autorización y procedimientos de validación.
  - Definir la estrategia de reorganización de las BD para aumentar la eficiencia del sistema.

1.4.1 MySQL

Es un sistema de gestión de base de datos relacional, multihilo y miultiusuario con más de seis millones de instalaciones que en los últimos años ha tenido un crecimiento vertiginoso. Es la base de datos de código abierto más popular del mundo. Código abierto significa que todo el mundo puede acceder al código fuente, es decir, al código de programación de MySQL, esto significa que también todos pueden contribuir con ideas, elementos, mejoras o sugerir optimizaciones. Y así es que este ha pasado de ser una pequeña base de datos a una completa herramienta. Su rápido desarrollo se debe en gran medida a la contribución de mucha gente al proyecto, así como la dedicación de su equipo.

MySQL incluye todos los elementos necesarios para instalar el programa, preparar diferentes niveles de acceso de usuario, administrar el sistema y proteger los datos. Utiliza el lenguaje de consulta estructurado (SQL). Es un sistema de administración de bases de datos relacional (RDBMS). Se trata de un programa capaz de almacenar una enorme cantidad de datos de gran variedad y de distribuirlos para cubrir las necesidades de cualquier tipo de organización, desde pequeños establecimientos comerciales a grandes empresas y organismos administrativos.

Antes se consideraba como la opción ideal de sitios web; sin embargo, ahora incorpora muchas de las funciones necesarias para otros entornos y conserva su gran velocidad. Es una base de datos robusta

que se la puede comparar con una base de datos comercial, es incluso más veloz en el procesamiento de las transacciones y dispone de un sistema de permisos elegante y potente, y ahora, además, además dispone de procedimientos de almacenado, triggers y vistas.

Es rápido, y una solución accesible para administrar correctamente los datos de una empresa, y, como ocurre con la mayor parte de las comunidades de código abierto, se puede encontrar una gran cantidad de ayuda en la Web.

Son muchas las razones para escoger a MySQL como una solución de misión crítica para la administración de datos: (23)

- **Costo:** Es gratuito para la mayor parte de los usos y su servicio de asistencia resulta económico.
- **Velocidad:** Es mucho más rápido que la mayoría de sus rivales.
- **Funcionalidad:** Dispone de muchas de las funciones que exigen los desarrolladores profesionales, como compatibilidad completa con ACID, compatibilidad para la mayor parte de SQL ANSI21, volcados online, duplicación, funciones SSL e integración con la mayor parte de los entornos de programación.
- **Portabilidad:** Se ejecuta en la inmensa mayoría de sistemas operativos y, la mayor parte de los casos, los datos se pueden transferir de un sistema a otro sin dificultad.
- **Facilidad de uso:** Resulta fácil de utilizar y de administrar. Las herramientas de MySQL son potentes y flexibles, sin sacrificar su capacidad de uso.

1.4.2 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional (ORDBMS) basado en el proyecto POSTGRES, de la universidad de Berkeley; es una derivación libre (OpenSource) de este proyecto, y utiliza el lenguaje SQL92/SQL99.

Fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido, más tarde en otros sistemas de gestión comerciales. Es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional.

Principales características de este gestor de bases de datos: (24)

- Implementación del estándar SQL92/SQL99.

- Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP) y cadenas de bits. También permite la creación de tipos propios.
- Incorpora una estructura de datos array.
- Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes.
- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta el uso de índices, reglas y vistas.
- Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.

PostgreSQL es un magnífico gestor de bases de datos, capaz de competir con muchos gestores comerciales, aunque carezca de alguna característica casi imprescindible, un conjunto de herramientas que permitan una fácil gestión de los usuarios y de las bases de datos que contenga el sistema. Por otro lado, la velocidad de respuesta que ofrece este gestor con bases de datos relativamente pequeñas puede parecer un poco deficiente, aunque esta misma velocidad la mantiene al gestionar bases de datos realmente grandes, cosa que resulta loable.

1.4.3 Microsoft SQL Server

Microsoft SQL Server es un sistema de gestión de bases de datos relacionales basado en el lenguaje Transact-SQL, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea.

Entre sus características figuran:

- Escalabilidad, estabilidad y seguridad.
- Soporta procedimientos almacenados.
- Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL (Data Definition Language) y DML(Data Manipulation Language) gráficamente.
- Permite trabajar en modo Cliente/Servidor donde la información y datos se alojan en el servidor y las terminales o clientes de la red sólo acceden a la información.
- Además permite administrar información de otros servidores de datos.

1.5 Framework

En el desarrollo de software, un framework, es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

Los frameworks son diseñados con el intento de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional.

Simplifican el desarrollo de las aplicaciones mediante la automatización de muchas de las tareas comunes y proporcionan estructura al código fuente, forzando al programador a crear código más legible y más fácil de mantener. (25)

Los objetivos principales que persigue un framework son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones. (26)

En general, el término framework, se refiere a una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, se puede considerar como una aplicación genérica incompleta y configurable a la que se le pueden añadir las últimas piezas para construir una aplicación concreta. (27)

1.5.1 CodeIgniter

CodeIgniter es un Framework para el desarrollo de aplicaciones para personas que quieran construir sitios usando PHP. Su objetivo es poder hacer los proyectos más rápidos de lo que se pueden hacer, suministrando un rico conjunto de librerías para tareas comunes, con una simple interfaz y estructura lógica para acceder a estas librerías. CodeIgniter deja que su creatividad se centre en el proyecto minimizando la cantidad de código necesitado para ejecutar una tarea. (28)

Está liberado bajo la licencia de código abierto Apache-BSD, lo que significa que es totalmente libre y puede ser usado. (29)

Entre sus características se puede encontrar su compatibilidad con PHP 4 y PHP 5, posee soporte para múltiples bases de datos entre las que se pueden mencionar PostgresSQL, MySQL, MSSQL, además de que se lo pueden incluir drivers para otros gestores de bases de datos que se programen, incluye también plantillas, validaciones, no requiere instalación, se pueden encontrar una librería con un gran número de clases. Una de las características más interesantes de CodeIgniter es el elevado número de clases que incluye para trabajar con distintos objetos: calendario, bases de datos, correo electrónico, manipulación de imágenes, FTP, lenguaje, tablas, sesiones y compresión ZIP.

A diferencia de otros frameworks, CodeIgniter cuenta con una documentación excelente que permite conocer todos los secretos de este entorno de trabajo. Requiere sólo de unas bibliotecas muy pequeñas. Esto está en contraste severo a muchos frameworks que requieren significativamente más recursos. Las bibliotecas adicionales serán dinámicamente cargadas según la demanda, basada en sus necesidades por un proceso dado, por lo que es bastante rápido. (30)

Puede extenderse fácilmente a través del uso de pluggins y bibliotecas de ayuda, o a través de extensiones de clases.

Está basado en el patrón Modelo Vista Controlador (MVC), donde las modelos representan las estructuras de datos y contienen las funciones para insertar y actualizar la base de datos; la vista es la información que es presentada al usuario, normalmente las vistas son páginas web y por último las controladoras son las intermediarias entre las modelos, las vistas y cualquier otro recurso que se necesite para procesar el pedido HTTP y generar la página web. (31)

Muchos de los Framework de PHP ofrecen MVC, pero ninguno es tan ligero y flexible como CodeIgniter. Lo mejor de todo es que este puede ser tan sólo VC (Vista Controlador), es decir no fuerza al usuario a utilizar una BD para un desarrollo. Cabe resaltar que su soporte de Ajax aunque mínimo es lo suficiente para desarrollar aplicaciones vistosas y ágiles y se podría decir sin lugar a dudas que CodeIgniter brinda una muy buena implementación del patrón Active Record, independizando al usuario o programador del sistema en el que esté la BD brindando para ello una amplia gama de funciones que abstraen de sentencias SQL explícitas y evitan las famosas SQL-Inyections.

1.5.2 Symfony

Symfony es un framework para construir aplicaciones web con PHP. En otras palabras, es un enorme conjunto de herramientas y utilidades que simplifican el desarrollo de las aplicaciones web. (32)

Es uno de los frameworks PHP más populares entre los usuarios y las empresas, ya que permite que los programadores sean mucho más productivos a la vez que crean código de más calidad y más fácil de mantener. Symfony es maduro, estable, profesional y está muy bien documentado. (33)

Emplea el tradicional patrón de diseño MVC para separar las distintas partes que forman una aplicación web. El modelo representa la información con la que trabaja la aplicación y se encarga de acceder a los datos. (34)

La vista transforma la información obtenida por el modelo en las páginas web a las que acceden los usuarios. El controlador es el encargado de coordinar todos los demás elementos y transformar las peticiones del usuario en operaciones sobre el modelo y la vista. (35)

Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix y Linux) como en plataformas Windows.

1.5.3 Yahoo user Interface (YUI)

Yahoo user Interface (YUI) Library es un conjunto de utilidades y controles, escrito en JavaScript para desarrollar aplicaciones Web interactivas (RIA - Rich Internet Application) usando técnicas DOM (Document Object Model), DHTML (Dynamic HTML) y AJAX (Asynchronous JavaScript And XML).YUI incluye también un conjunto de fuentes CSS (Cascading Styles Sheets). Todos los componentes YUI han sido desarrollados bajo Licencia BSD Open Source y están disponibles para todo tipo de uso. (36)

Algunas características:

- Están disponibles dos tipos de componentes diferentes: Utilidades y Controles.
- Las YUI Utilidades simplifican el desarrollo para la compatibilidad entre Navegadores basados en técnicas DOM, DHTML y AJAX.
- Los controles de YUI proporcionan elementos visuales altamente interactivos del diseño para sus aplicaciones Web. Estos elementos se crean y se manejan íntegramente del lado del cliente (usuario) y nunca requieren de una recarga de página.

1.6 Entorno Integrado de Desarrollo (IDE)

Un entorno de desarrollo integrado o en inglés Integrated Development Environment ('IDE') es un programa compuesto por un conjunto de herramientas para un programador. (37)

Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios. (38)

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDEs pueden ser aplicaciones por si solas o pueden ser parte de aplicaciones existentes. (39)

Componentes de un IDE: (40)

- Un editor de texto. Un compilador.
- Un intérprete.
- Herramientas de automatización.
- Un depurador.
- Posibilidad de ofrecer un sistema de control de versiones.
- Factibilidad para ayudar en la construcción de interfaces gráficas de usuarios.

1.6.1 Zend Studio

Editor web orientado a la programación de páginas PHP, con ayudas en la gestión de proyectos y depuración de código.

Los expertos en PHP consideran a Zend Studio como el entorno IDE más maduro y con más características útiles. La última versión ofrece manipulación avanzada de bases de datos y otras mejoras. Se trata de un programa de la casa Zend, impulsores de la tecnología de servidor PHP, orientada a desarrollar aplicaciones web, en lenguaje PHP. (41)

El programa, además de servir de editor de texto para páginas PHP, proporciona una serie de ayudas que pasan desde la creación y gestión de proyectos hasta la depuración de código.

El programa entero está escrito en Java, lo que a veces supone que no funcione tan rápido como otras aplicaciones de uso diario. Sin embargo, esto ha permitido a Zend lanzar con relativa facilidad y rapidez versiones del producto para Windows, Linux y MacOS, aunque el desarrollo de las versiones de este último sistema se retrase un poco más.

Zend Studio consta de dos partes en las que se dividen las funcionalidades de parte del cliente y las del servidor. Las dos partes se instalan por separado, la del cliente contiene el interfaz de edición y la ayuda. Permite además hacer depuraciones simples de scripts, aunque para disfrutar de toda la potencia de la herramienta de depuración habrá que disponer de la parte del servidor, que instala Apache y el módulo PHP o, en caso de que estén instalados, los configura para trabajar juntos en depuración. (42)

Lo más destacable es que contiene una ayuda contextual con todas las librerías de funciones del lenguaje que asiste en todo momento ofreciendo nombres de las funciones y parámetros que deben recibir. Aunque esta ayuda contextual no solo se queda en las funciones definidas en el lenguaje, sino que también reporta ayudas con las funciones que se vayan creando, incluso en páginas que se tengan incluidas con la función include().

Otras ayudas que ofrece a la hora de escribir son las típicas en editores avanzados, como permitir editar varios archivos, y moverse fácilmente entre ellos, marcar a qué elementos corresponden los inicios y cierres de las etiquetas, paréntesis o llaves, moverse al principio o al final de una función e identificación automática del código.

Sin duda más de una vez los programadores de PHP se han visto en un duro problema por no encontrar un error en algún script que está haciendo devuelva resultados inesperados. En estos casos lo que se suele hacer es escribir el contenido de diversas variables en la página web para que le den al

programador algún indicio del lugar donde está el error. Para contrarrestar esto Zend Studio dispone de una herramienta muy interesante de depuración. Gracias a ella se pueden ejecutar páginas y conocer en todo momento el contenido de las variables de la aplicación y las variables del entorno como las cookies, las recibidas por formulario o en la sesión. Se pueden colocar puntos de parada de los scripts y realizar las acciones típicas de depuración.

Además de la ventana para visualizar el contenido de las variables, dispone de otras donde muestra la salida script según se va generando, y otra donde se pueden ver las alertas y errores. (43)

1.6.2 Notepad++

Es un poderoso editor de código fuente que soporta diversos lenguajes de programación. Permite imprimir el código escrito con los diferentes colores de la sintaxis, posee la opción de auto completado con la posibilidad de definir una API personalizada, permite la edición de varios documentos al mismo tiempo, con sus correspondientes vistas separadas o integradas a la misma interfaz.

Notepad++ posee soporte para búsqueda y reemplazo de expresiones regulares, detección automática del estado del texto, incluye una herramienta de Zoom, trabaja con puntos de marca para editar y desplazarse más dinámicamente y permite la creación de marcos con su correspondiente secuencia de teclas para acceso rápido. (44)

Con Notepad++ se puede programar en los siguientes lenguajes: C, C++, Java, C#, XML, HTML, PHP, CSS, makefile, ASCII art (.nfo), doxygen, ini file, batch file, Javascript, ASP, VB/VBS, SQL, Objective-C, RC resource file, Pascal, Perl, pitón, Lua, TeX, TCL, Assembler, Ruby, Lisp, Scheme, Properties, Diff, Smalltalk, Postscript, VHDL, Ada, Caml, AutoIt, KiXtart, Matlab, Verilog, Haskell, InnoSetup y CMake.

1.7 Tecnologías y herramientas a utilizar

El Módulo Vectores del proyecto Control Sanitario Internacional basa su realización en un conjunto de herramientas definidas por el Grupo de Arquitectura MINSAP-MIC para el desarrollo de todas las aplicaciones para INFOMED. Estas están conformadas por el servidor Web Apache 2.0, el cual es un servidor libre de código abierto y bajo la licencia apache/GPL, como lenguaje de programación define PHP en su más reciente versión, la 5.0, y como gestor de bases de datos el ya conocido y libre MySQL, igualmente en la versión 5.

Se utilizará además el IDE de programación Zend Studio por las grandes prestaciones que este brinda y para la implementación del sistema se hará uso del framework CodeIgniter, el cual fue previamente definido por el Área Temática Sistemas Especializados para la Salud de la Facultad 7 de la Universidad de las Ciencias Informáticas para llevar a cabo el desarrollo de las aplicaciones que se implementen en esta área.

Conclusiones

Se realizó un análisis de las tendencias y tecnologías actuales más utilizadas para el desarrollo de aplicaciones web; así como, de las herramientas predefinidas por el Grupo de Arquitectura MINSAP-MIC y el Área Temática Sistemas Especializados para la Salud de la Facultad 7 de la Universidad de las Ciencias Informáticas para el desarrollo de la aplicación, con vistas a desarrollar un sistema integrado a toda una infraestructura que se ha venido desarrollando con la finalidad de constituir la base informatizada de la salud pública cubana, teniendo en cuenta las políticas de Cuba determinadas por el uso de herramientas gratuitas y bajo licencias de software libre para el desarrollo de sus aplicaciones.

Todo esto con el objetivo de profundizar en el conocimiento de estos temas y fundamentar el uso de las tecnologías y herramientas utilizadas para llevar a cabo la solución informática a la situación existente.

Capítulo 2: Elementos de Arquitectura

Introducción

En el siguiente capítulo se expone la arquitectura que ha sido determinada para el desarrollo de la aplicación de Vectores, mostrando para ello los requisitos no funcionales, los patrones arquitectónicos seguidos en el desarrollo de la aplicación y una explicación de los modelos de despliegue y de implementación.

2.1 Arquitectura de Software

Una Arquitectura de Software, también denominada Arquitectura lógica, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información.

La Arquitectura de Software establece los fundamentos para que analistas, diseñadores y programadores trabajen en una línea común que permita alcanzar los objetivos del sistema de información.

Se selecciona y diseña con base en objetivos y restricciones. Los objetivos son aquellos prefijados para el sistema de información, pero no solamente los de tipo funcional, también otros objetivos como la mantenibilidad, auditabilidad, flexibilidad e interacción con otros sistemas de información. Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información. Unas arquitecturas son más recomendables de implementar con ciertas tecnologías mientras que otras tecnologías no son aptas para determinadas arquitecturas.

La arquitectura de software define, de manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación ente ellos. Toda arquitectura debe ser implementable en una arquitectura física, que consiste simplemente en determinar qué computadora tendrá asignada cada tarea.

La arquitectura de software, tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales, como la confiabilidad, escalabilidad, portabilidad, y disponibilidad.

En fin, la Arquitectura de Software representa un alto nivel de abstracción común que la mayoría de los participantes, si no todos, pueden usar como base para crear entendimiento mutuo, formar consenso y comunicarse entre sí. En sus mejores expresiones, la descripción arquitectónica expone las restricciones de alto nivel sobre el diseño del sistema, así como la justificación de decisiones arquitectónicas fundamentales. (45)

Tanto en la arquitectura como en los patrones, la idea dominante es la de reutilización.

2.2 Requerimientos No Funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. En muchos casos los requerimientos no funcionales son fundamentales en el éxito del producto.

Normalmente están vinculados a requerimientos funcionales, es decir, una vez se conozca lo que el sistema debe hacer se puede determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser.

Los requerimientos no funcionales forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, pues si se conoce que el mismo cumple con toda la funcionalidad requerida, las propiedades no funcionales, como cuán usable, seguro, conveniente y agradable, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación.

A continuación se muestran los requerimientos no funcionales correspondientes al sistema en cuestión:

2.2.1 Requerimientos de Apariencia o Interfaz Externa

- Se podrán distinguir colores atractivos y acordes con los recomendados para los software de salud.
- Debe poseer un ambiente amigable, intuitivo, sencillo y de fácil navegación, tratando así de impedir el rechazo por parte del usuario al tener que interactuar con un sistema no conocido.
- Paginación de reportes de búsqueda, y listados.

- Diseño perfectamente encuadrado para resoluciones de 1024 x 768, pero preparado para verse en otras resoluciones.

2.2.2 Requerimientos de Usabilidad

- La aplicación Web debe ser flexible y de fácil aprendizaje, pues se trata en todo lo posible de mantener un estándar de operabilidad que logre que las interacciones del usuario con el sistema sean predecibles y familiares.
- El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora y de un ambiente Web en sentido general.

2.2.3 Requerimientos de Soporte

- El sistema contará con una ayuda para facilitar al usuario el manejo de la aplicación.
- Estará bien documentado para garantizar futuros mantenimientos.

2.2.4 Requerimientos de Portabilidad

- El sistema podrá ejecutarse sobre plataforma Linux, Windows 98 o superior.

2.2.5 Requerimientos de Seguridad.

- La autenticación de los usuarios en el sistema será garantizada por el Sistema de Autenticación, Autorización y Auditoría (SAAA) al cual estará conectada la aplicación.
- Las funcionalidades deberán estar restringidas de acuerdo a los diferentes roles con el objetivo de garantizar que el acceso a la información sea según el nivel de autorización.
- El sistema deberá estar protegido ante el acceso no autorizado a la información.

2.2.6 Requerimientos de Software

Del lado del cliente:

- El cliente podrá tener acceso al sistema a través de los navegadores Web Mozilla Firefox o Internet Explorer.
- Sistema operativo Linux o Windows 98 ó Superior

Del lado del servidor:

- Sistema operativo Linux.

- Servidor Web Apache 2.0 y PHP 5.
- Servidor de Base de Datos MySQL 5.1.
- Framework CodeIgniter.

2.2.7 Requerimientos de Hardware

Del lado del cliente:

- Procesador Pentium III o superior.
- 256 de memoria RAM o superior.
- Monitor VGA o superior.
- Tarjeta de red.

Del lado del servidor:

- Procesador Pentium IV o superior.
- 2GB de memoria RAM o superior.
- Disco Duro de 80 GB.
- Monitor tipo VGA o superior.
- Tarjeta de red.

2.2.8 Requerimientos de Diseño e Implementación

- Se utilizarán los patrones de diseño GRASP.
- Se utilizará como metodología de desarrollo RUP (Rational Unified Process), como lenguaje de modelado UML (Unified Modeling Language) y como herramienta case el Visual Paradigm.
- Para la implementación se utilizará como lenguaje de programación PHP y como IDE de desarrollo Zend Studio.

2.2.9 Requerimientos de Ayuda y Documentación en línea

- Contará con un manual de usuario para que se pueda explotar al máximo.
- Contará con una ayuda digital, a la cual se podrá acceder desde cualquier parte de la aplicación.

2.3 Estilos o patrones arquitectónicos presentes en la solución. Características y justificación de su uso

2.3.1 Modelo Cliente/Servidor

La arquitectura Cliente/Servidor es un modelo para el desarrollo de sistemas de información en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos y servidor al proceso que responde a las solicitudes.

Los clientes realizan generalmente funciones como:

- Manejo de la interfaz de usuario.
- Captura y validación de los datos de entrada.
- Generación de consultas e informes sobre las bases de datos.

Por su parte los servidores realizan las siguientes funciones:

- Gestión de periféricos compartidos.
- Control de accesos concurrentes a bases de datos compartidas.
- Enlaces de comunicaciones con otras redes de área local o extensa.

Siempre que un cliente requiere un servicio lo solicita al servidor correspondiente y éste le responde proporcionándolo. Normalmente, pero no necesariamente, el cliente y el servidor están ubicados en distintos procesadores. Los clientes se suelen situar en ordenadores personales y/o estaciones de trabajo y los servidores en procesadores departamentales o de grupo.

Entre las principales características de la arquitectura Cliente/Servidor se pueden destacar las siguientes: [33]

- El servidor presenta a todos sus clientes una interfaz única y bien definida.
- El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.
- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- Los cambios en el servidor implican pocos o ningún cambio en el cliente.

Ventajas:

- Menores costes de operación:
  - Permiten un mejor aprovechamiento de los sistemas existentes, protegiendo la inversión. Por ejemplo, la compartición de servidores (habitualmente caros) y dispositivos periféricos (como impresoras) entre máquinas clientes permite un mejor rendimiento del conjunto.
  - Proporcionan un mejor acceso a los datos. La interfaz de usuario ofrece una forma homogénea de ver el sistema, independientemente de los cambios o actualizaciones que se produzcan en él y de la ubicación de la información.
  - El movimiento de funciones desde un ordenador central hacia servidores o clientes locales origina el desplazamiento de los costes de ese proceso hacia máquinas más pequeñas y por tanto, más baratas.

- Mejora en el rendimiento de la red:
  - La arquitectura Cliente/Servidor elimina la necesidad de mover grandes bloques de información por la red hacia los ordenadores personales o estaciones de trabajo para su proceso. Los servidores controlan los datos, procesan peticiones y después transfieren sólo los datos requeridos a la máquina cliente. Entonces, la máquina cliente presenta los datos al usuario mediante interfaces amigables. Todo esto reduce el tráfico de la red, lo que facilita que pueda soportar un mayor número de usuarios.
  - Tanto el cliente como el servidor pueden escalarse para ajustarse a las necesidades de las aplicaciones.
  - En una arquitectura como ésta, los clientes y los servidores son independientes los unos de los otros con lo que pueden renovarse para aumentar sus funciones y capacidad de forma independiente, sin afectar al resto del sistema.

En el caso del módulo Vectores se utiliza este modelo ya que la aplicación estará ubicada en un servidor central en INFOMED, el cual se encargará de dar respuesta a las peticiones realizadas por los clientes que utilicen la aplicación.

2.3.2 Arquitectura en capas

El modelo actual de desarrollo ha demostrado que organizar los elementos de las aplicaciones en componentes independientes puede lograr una mayor eficiencia durante el tiempo de desarrollo y mantenimiento. La programación en múltiples capas es la técnica más efectiva para aplicaciones

empresariales, dividir los componentes de la aplicación en capas implica una fácil administración y rapidez en entornos Cliente/Servidor. Esta arquitectura consiste en dividir los componentes primarios de una aplicación, programarlos por separado y después unirlos, ya sea en tiempo de diseño o de ejecución.

Las aplicaciones Web se modelan mediante lo que se conoce como arquitecturas de capas:
Definición: Una capa representa un elemento del sistema que procesa o trata la información.
Características: Una capa puede residir (se ejecuta) en una máquina diferente o en diferentes espacios o entornos de proceso dentro de la misma máquina.

Existen varias razones que inclinan a utilizar esta arquitectura en el desarrollo de aplicaciones:

- Abstracción total acerca del origen de datos.
  Las distintas capas se especializan solamente en la funcionalidad que deben brindar (procesamiento de las reglas del negocio, presentación de los datos y acceso a los mismos) sin importar cual es el origen de los datos procesados.
- Bajo costo de desarrollo y mantenimiento de las aplicaciones.
  La utilización de esta arquitectura deja observar al momento de diseño una mayor carga en complejidad, sin embargo, la utilización de la misma brinda un control más cercano de cada componente, así como también la posibilidad de una verdadera reutilización de código. La modificación de una de las capas puede realizarse con un mínimo de impacto en las demás.
- Estandarización de las reglas del negocio.
  Las reglas del negocio se encuentran en una librería común y pueden ser llamadas desde diversas aplicaciones sin saber como funciona o ha sido diseñada.
- Mejor calidad de las aplicaciones.
  Como las aplicaciones son construidas en unidades separadas, estas pueden ser probadas independientemente y con mucho más detalle, el desarrollador (o los desarrolladores) solo debe preocuparse por el funcionamiento de la capa que están desarrollando, esto conduce a obtener un producto mucho más sólido.
- Mejor funcionamiento de las aplicaciones.
  Las capas pueden ejecutarse en máquinas independientes, que se traduce en mejor tiempo de respuesta a los usuarios.

La arquitectura más utilizada es la arquitectura de tres capas. Separa los datos de una aplicación, la interfaz de usuario y la lógica en tres componentes distintos.

Capítulo 2: Elementos de Arquitectura

Capas o niveles

- **Capa de presentación**

Es la capa que ve el usuario, presenta el sistema al usuario, le comunica la información y captura la información de este dando un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio.

La capa de Presentación provee su aplicación con una interfaz de usuario (IU). Aquí es donde la aplicación presenta información a los usuarios y acepta entradas o respuestas del usuario para usar por el programa. Idealmente, la interfaz de usuario no desarrolla ningún procesamiento de negocios o reglas de validación de negocios. Por el contrario, la IU debería relegar sobre la capa de negocios para manipular estos asuntos. Esto es importante, especialmente hoy en día, debido a que es muy común para una aplicación tener múltiples IU, o para sus clientes o usuarios, que le solicitan que elimine una IU y la remplace con otra.

- **Capa de negocio**

Es donde reside la lógica de funcionamiento de la aplicación o lógica del negocio, recibiendo las peticiones del usuario y enviando las respuestas tras el proceso. Se denomina capa de negocio pues es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos recuperación o almacenamiento de datos.

- **Capa de datos**

Es donde residen los datos. Está formada por uno o más gestores de base de datos que realizan todo el almacenamiento de los mismos, reciben solicitudes de almacenamiento y recuperación de información desde la capa de negocio.

Todas estas capas pueden residir en un único servidor, pero lo más usual es que las capas estén en ordenadores diferentes o agrupadas en dependencia de las necesidades de la empresa o de la aplicación.

Ventajas de una aplicación tres capas

- Las llamadas de la interfaz del usuario en la estación de trabajo, al servidor de capa intermedia, son más flexibles que en el diseño de dos capas, ya que la estación sólo necesita transferir parámetros a la capa intermedia.

- Con la arquitectura de tres capas, la interfaz del cliente no es requerida para comprender o comunicarse con el receptor de los datos. Por lo tanto, esa estructura de los datos puede ser modificada sin cambiar la interfaz del usuario en la PC.
- El código de la capa intermedia puede ser reutilizado por múltiples aplicaciones si está diseñado en formato modular. Esto puede reducir los esfuerzos de desarrollo y mantenimiento, así como los costos de migración.
- La separación de roles en tres capas, hace más fácil reemplazar o modificar una capa sin afectar a los módulos restantes.
- Separando la aplicación de la base de datos, hace más fácil utilizar nuevas tecnologías de agrupamiento y balance de cargas.
- Separando la interfaz del usuario de la aplicación, libera de gran procesamiento a la estación de trabajo y permite que las actualizaciones de la aplicación sean centralizadas en el servidor de aplicaciones.

En el caso del módulo Vectores se utiliza está arquitectura debido a que se logra separar la lógica de presentación de la lógica del negocio, aunque esta última se encuentra un tanto acoplada a la lógica de acceso a datos, sin embargo se logra una gran abstracción en cuanto al Sistema Gestor de Base Datos a utilizar por lo que si este es reemplazado, el impacto sobre la lógica del acceso a datos será mínimo.

2.3.3 Arquitectura basada en componentes

La complejidad de los sistemas computacionales actuales ha llevado a buscar la reutilización del software existente. El desarrollo de software basado en componentes permite reutilizar piezas de código pre elaborado que permiten realizar diversas tareas, conllevando a diversos beneficios como las mejoras en la calidad, la reducción del ciclo de desarrollo. (46)

El objetivo fundamental de la Arquitectura Basada en Componentes es construir aplicaciones ensamblando módulos o componentes que han sido anteriormente diseñados por un grupo de personas, con el fin de que puedan ser rehusados en múltiples aplicaciones.

El uso de este paradigma posee algunas ventajas: (47)

- Reutilización del software.

- Simplifica las pruebas al permitir que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados.
- Simplifica el mantenimiento del sistema, pues cuando existe un débil acoplamiento entre componentes, el desarrollador es libre de actualizar y/o agregar componentes según sea necesario, sin afectar otras partes del sistema.
- Mayor calidad. Dado que un componente puede ser construido y luego mejorado continuamente por un experto u organización, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo.

En el caso del módulo Vectores se utiliza está arquitectura debido a que el sistema en sí mismo constituye un componente del Sistema de Información para la Salud y además se integra a otros componentes ya existentes en el mismo.

2.3.4 Modelo Vista Controlador (MVC)

Modelo Vista Controlador es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista.

El patrón de diseño software MVC establece que la arquitectura se divide en tres partes fundamentales:

Modelo: Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.

Vista: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

Controlador: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

En fin, en MVC el controlador es el encargado de redirigir un procesamiento determinado para cada petición que reciba. Por tanto, el controlador debe poseer algún modo de conocer la correspondencia entre peticiones y posibles respuestas; deberá tener un mapa de peticiones y respuestas. El modelo representa la aplicación que responde una petición: los datos y las reglas de negocio. Por último, la vista define la forma de mostrar la información al usuario. En el caso del patrón MVC, el procesamiento se lleva a cabo entre sus tres componentes. El controlador recibe una petición y decide quién la lleva a cabo en la capa del modelo. Una vez que el modelo termina las operaciones pertinentes, devuelve el control de ejecución al controlador, y éste envía los resultados a la capa de la vista para que muestre los resultados al usuario. (48)

Muchos sistemas informáticos utilizan un Sistema de Gestión de Base de Datos para gestionar los datos. En MVC corresponde al modelo.

En el caso del módulo de Vectores se utilizó este patrón ya que el framework utilizado (CodeIgniter) se basa en el mismo, permitiendo así separar la lógica del negocio y el acceso a datos de las interfaces de usuario, evitando que cuando uno de estos se viera afectado, influyera sobre los otros.

2.4 Modelo de Despliegue

Para el desarrollo de una aplicación informática es muy importante que el arquitecto de software preste especial atención a la configuración de hardware en la cual se desplegará el sistema, identificando nodos procesadores (computadoras), dispositivos y protocolos; todo lo cual en su conjunto conforma el modelo de despliegue, el cual define la arquitectura física del sistema por medio de nodos interconectados. Estos nodos son elementos hardware sobre los cuales pueden ejecutarse los elementos software y para representar esta información se emplea el elemento de UML denominado Diagrama de Despliegue.

Para el despliegue del Sistema Vectores del Proyecto Control Sanitario Internacional se contará con un Servidor de Aplicaciones donde se encontrará el código fuente de la aplicación, el mismo se comunicará mediante protocolo SOAP con el Servidor Web RIS donde se encontrarán los diferentes componentes del Sistema de Información para la Salud que se necesitarán utilizar para llevar a cabo el desarrollo del sistema de Vectores como parte de su propia arquitectura basada en componentes.

Además contará con un Servidor de Base de Datos donde estará la base da datos de la aplicación, el cual se comunicará mediante protocolo TCP/IP con el Servidor de Aplicaciones; y por último contará con las PC Cliente que serán las encargadas de permitirle al usuario el acceso al sistema mediante protocolo HTTP.

A continuación se presenta el Diagrama de Despliegue correspondiente al Módulo Vectores del Sistema de Control Sanitario Internacional.

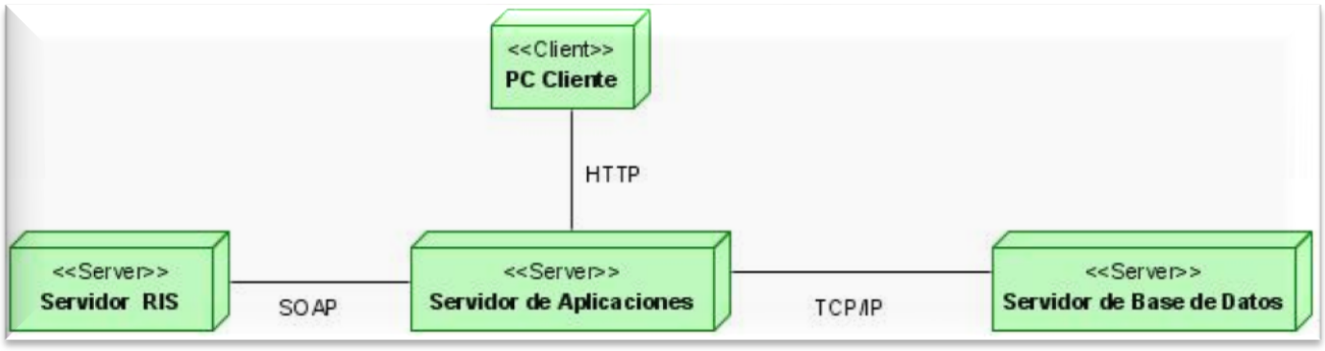

Figura 1. Diagrama de Despliegue.

Servidor RIS: En este nodo se encuentran desplegados los componentes del RIS con los que debe interactuar el sistema de Vectores.

Servidor de Aplicaciones: En este nodo se encuentra la aplicación de Vectores

.

Servidor de Base de Datos: En este nodo se encuentra el servidor de la BD del sistema Vectores.

PC Cliente: Desde este se accede a través de un navegador al sistema.

2.5 Modelo de Implementación

En la implementación se empieza con el resultado del diseño y se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, scripts y ejecutables, para lo cual se realiza el modelo de implementación, el cual incluye los componentes (que representan al código fuente) y la correspondencia de las clases con los componentes.

Para el desarrollo del modelo de implementación se dividió el sistema en subsistemas de implementación, que no son más que una colección de componentes y otros subsistemas de implementación usados para estructurar el modelo de implementación y dividirlos en pequeñas partes que pueden ser integradas y probadas de forma separada.

El subsistema Vectores está dividido en tres subsistemas de implementación fundamentales: el subsistema Vistas, el subsistema Controladoras y el subsistema Modelos, estructurados de modo que agrupan las clases según el rol que desempeñan dentro del patrón arquitectónico Modelo-Vista-Controlador. Además cuenta con un subsistema externo que contiene los módulos del RIS con los que este interactúa y la base de datos que será la encargada de la manipulación de la información.

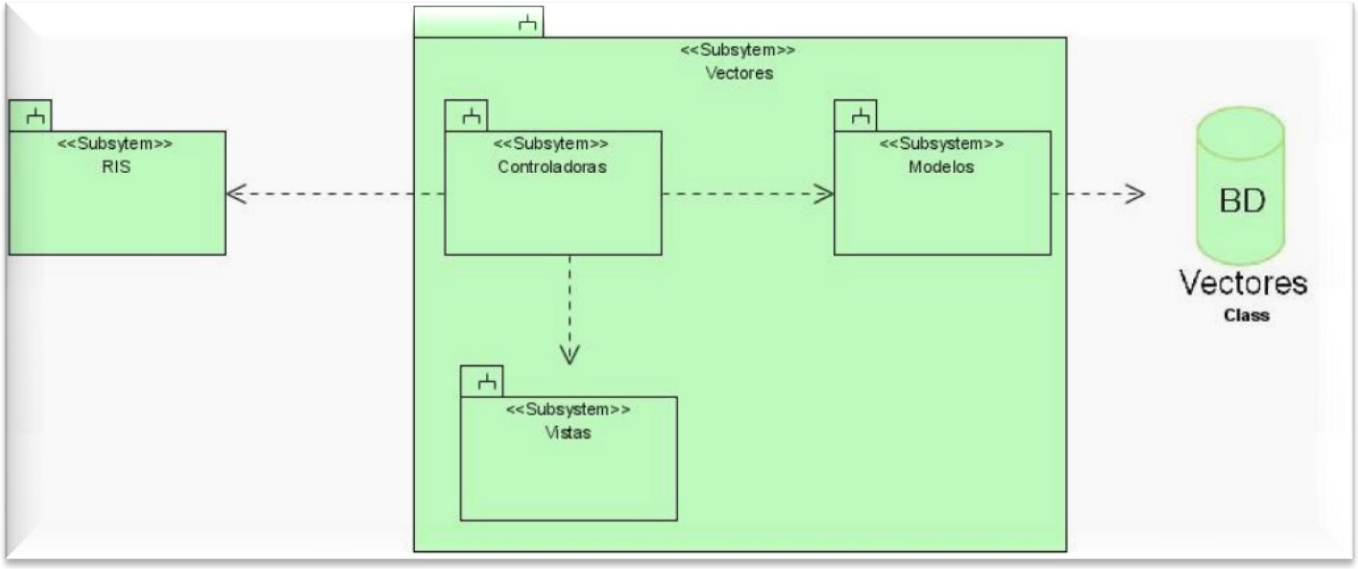

Figura 2. Diagrama de componentes.

A continuación se muestran los diferentes subsistemas en un mayor grado de detalle.

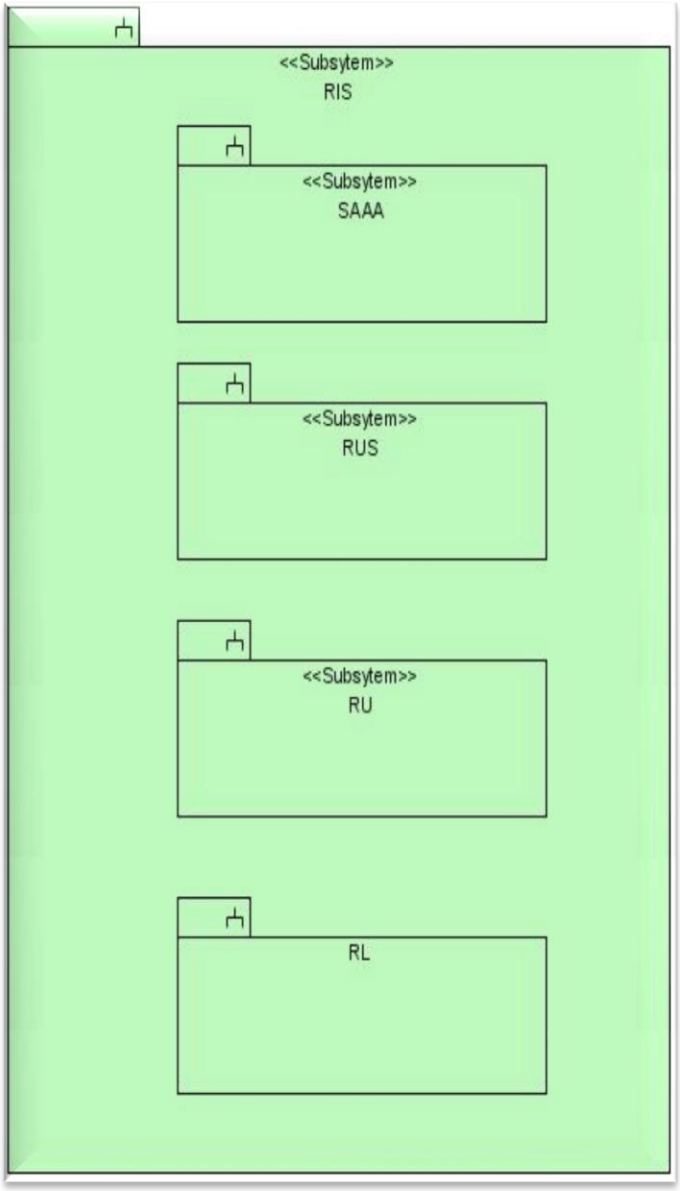

Figura 3. Subsistema RIS.

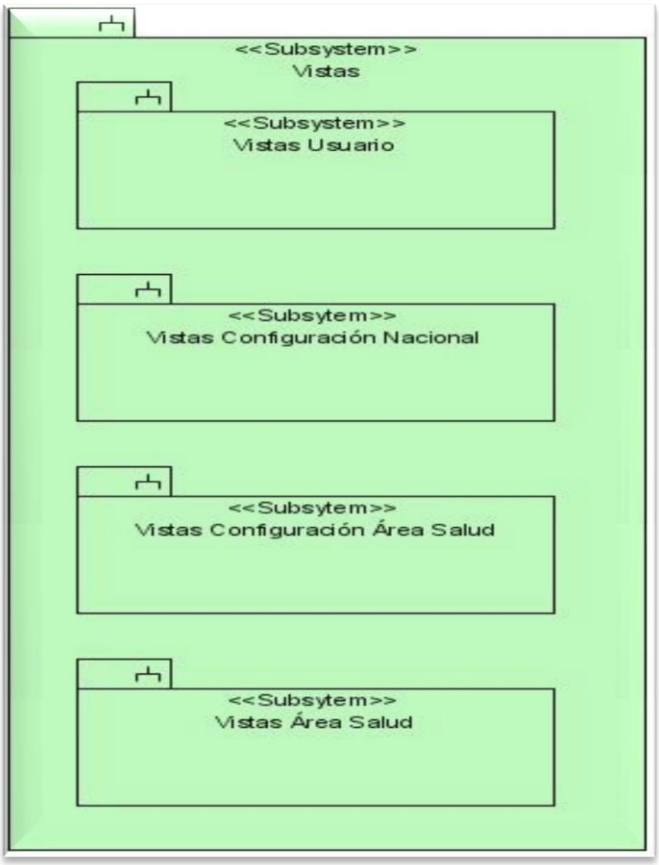

Figura 4. Subsistema vistas.

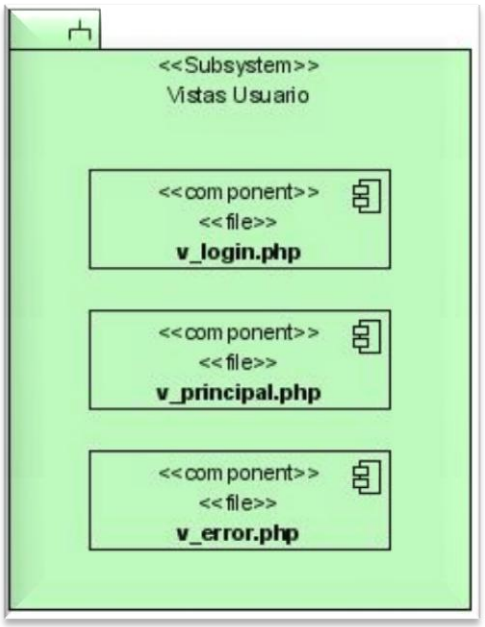

Figura 5. Subsistema Vistas Usuario.

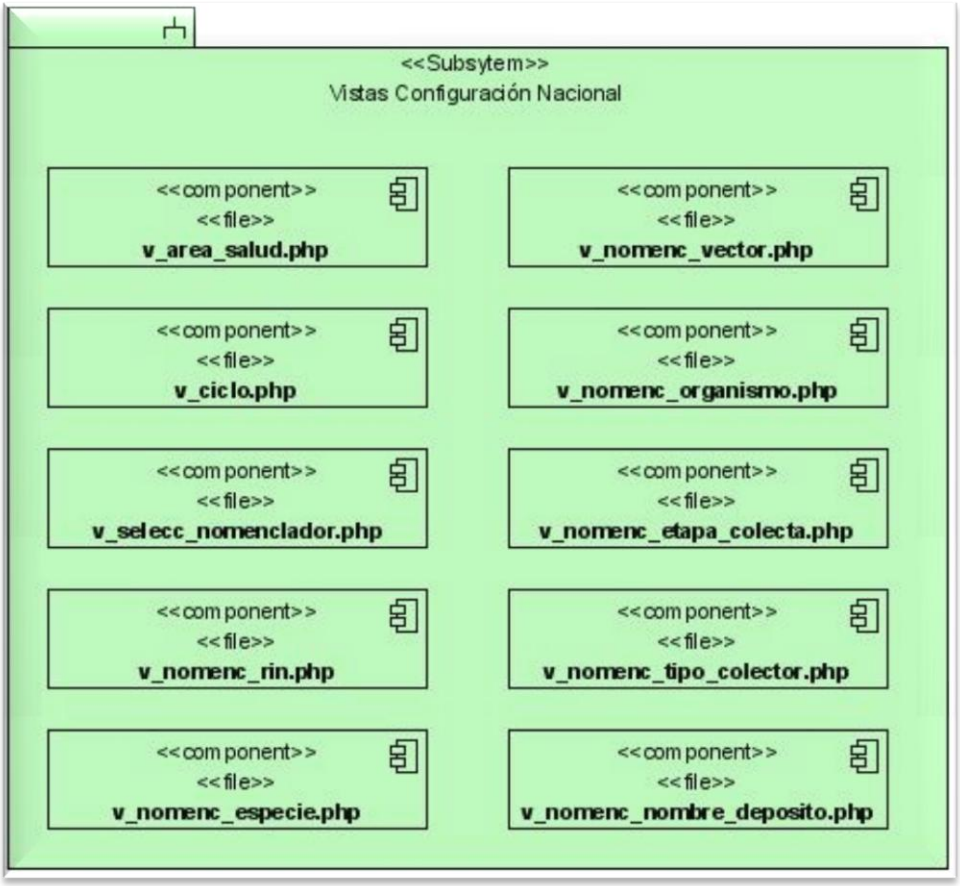

Figura 6. Subsistema Vistas Configuración Nacional.

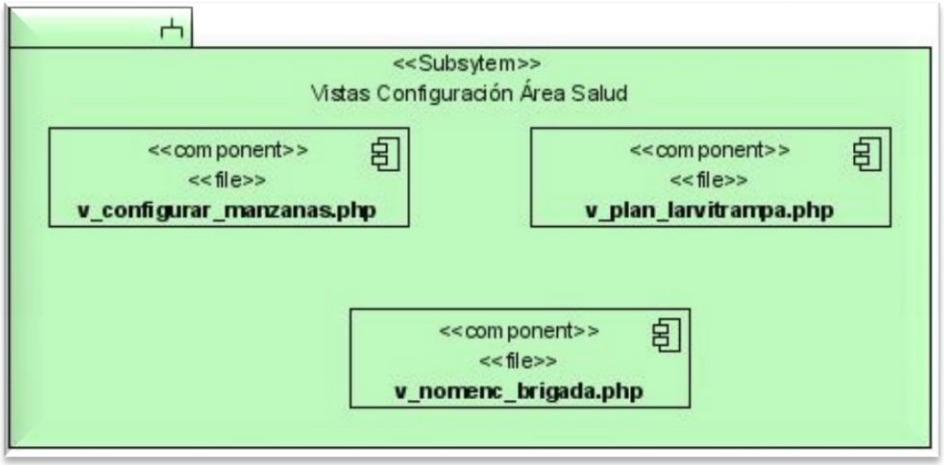

Figura 7. Subsistema Vistas Configuración área salud.

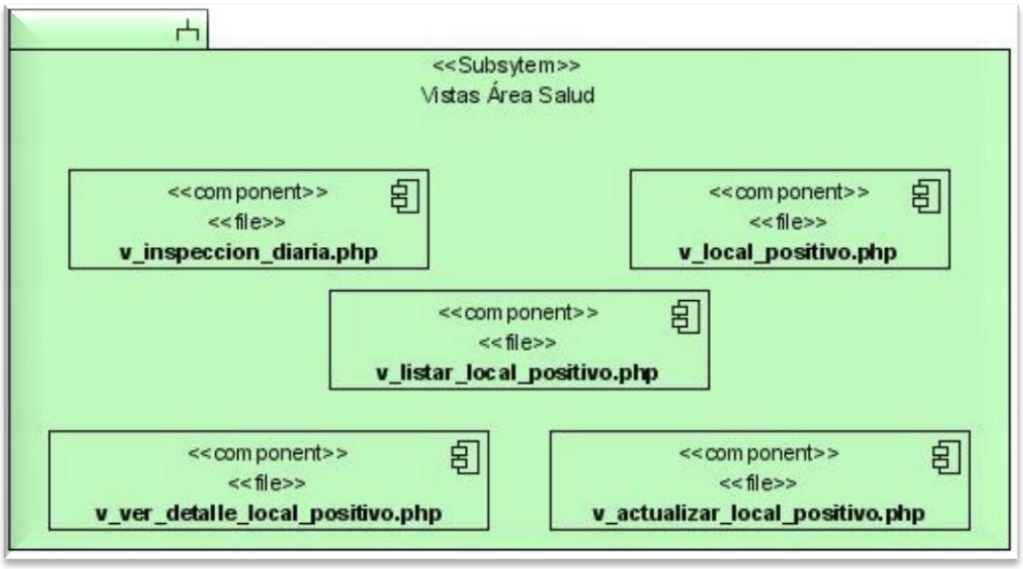

Figura 8. Subsistema Vistas Área Salud.

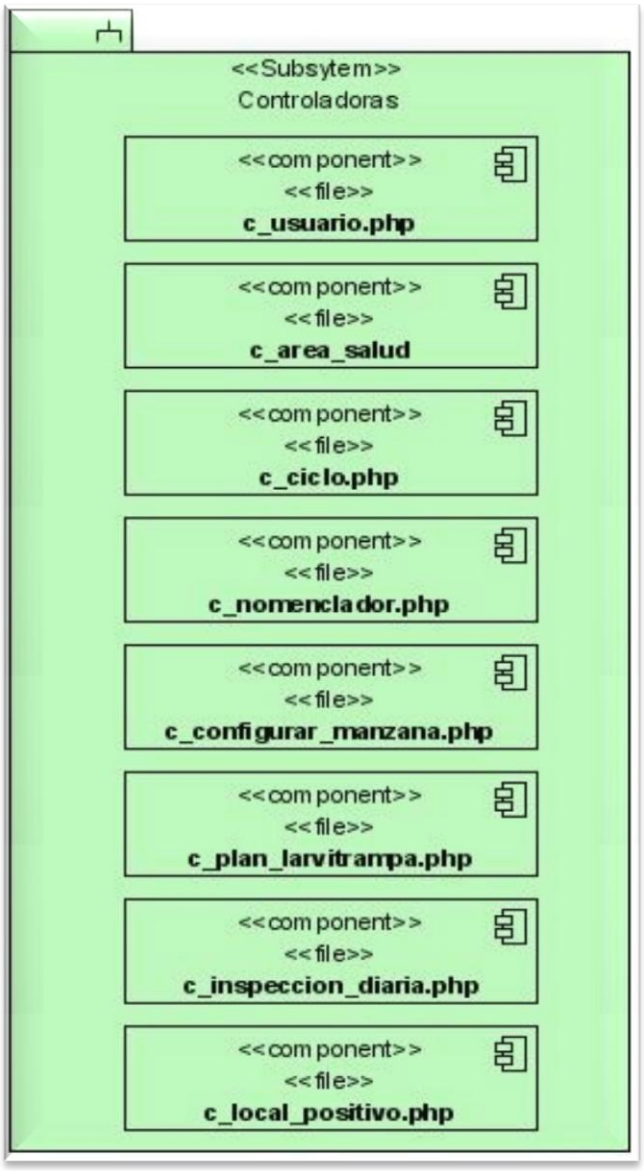

Figura 9. Subsistema Controladoras.

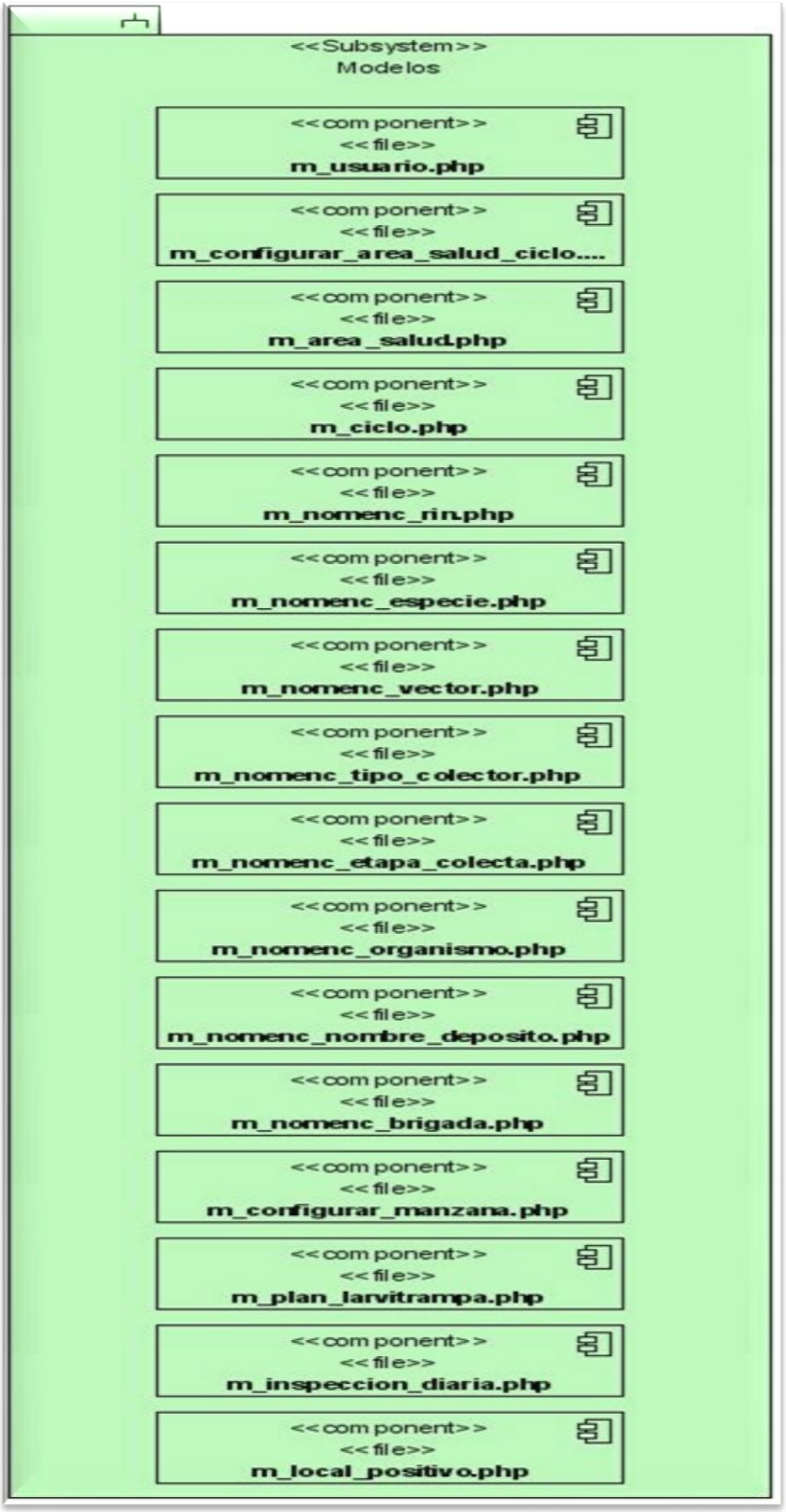

Figura 10. Subsistema Modelos.

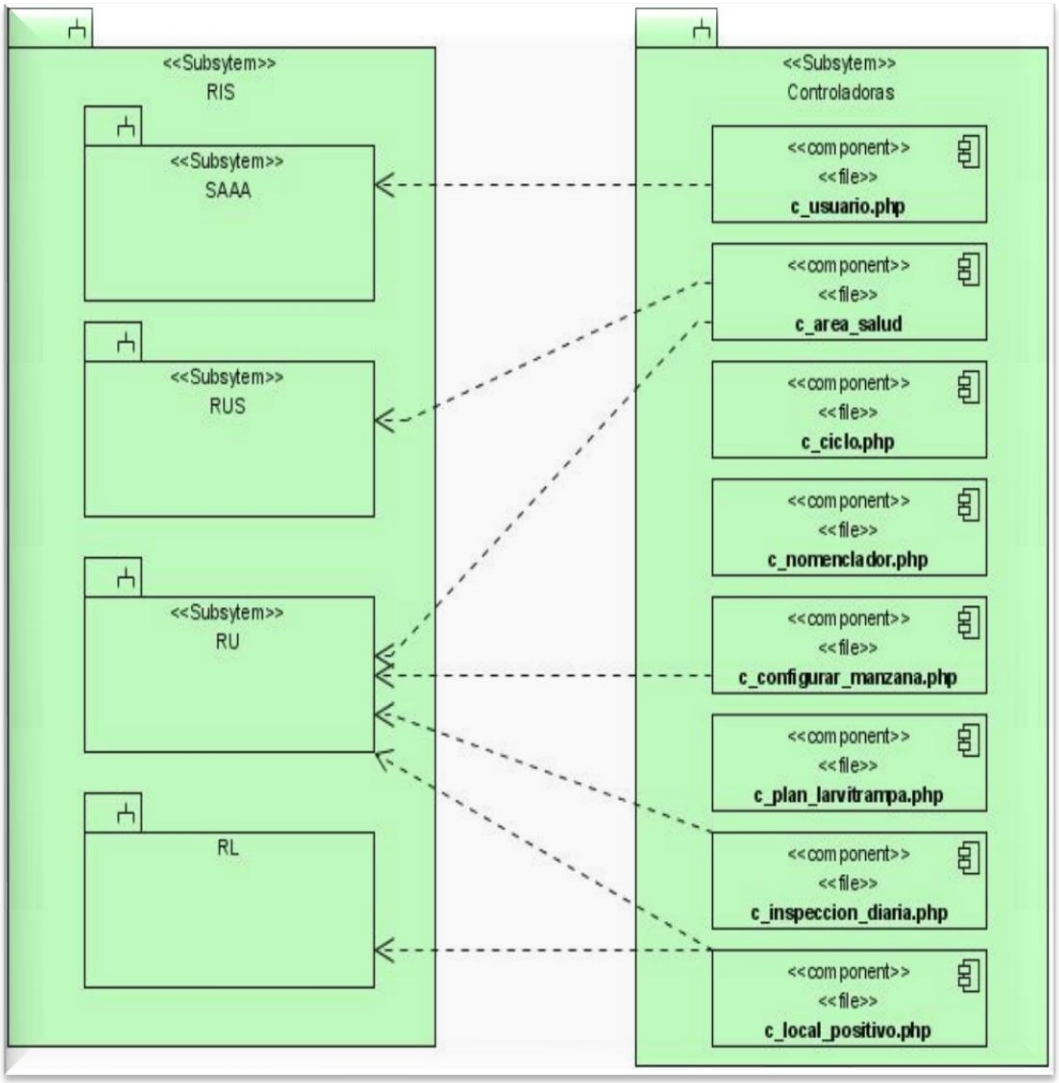

Figura 11. Relación entre el Subsistema RIS y el Subsistema Controladoras.

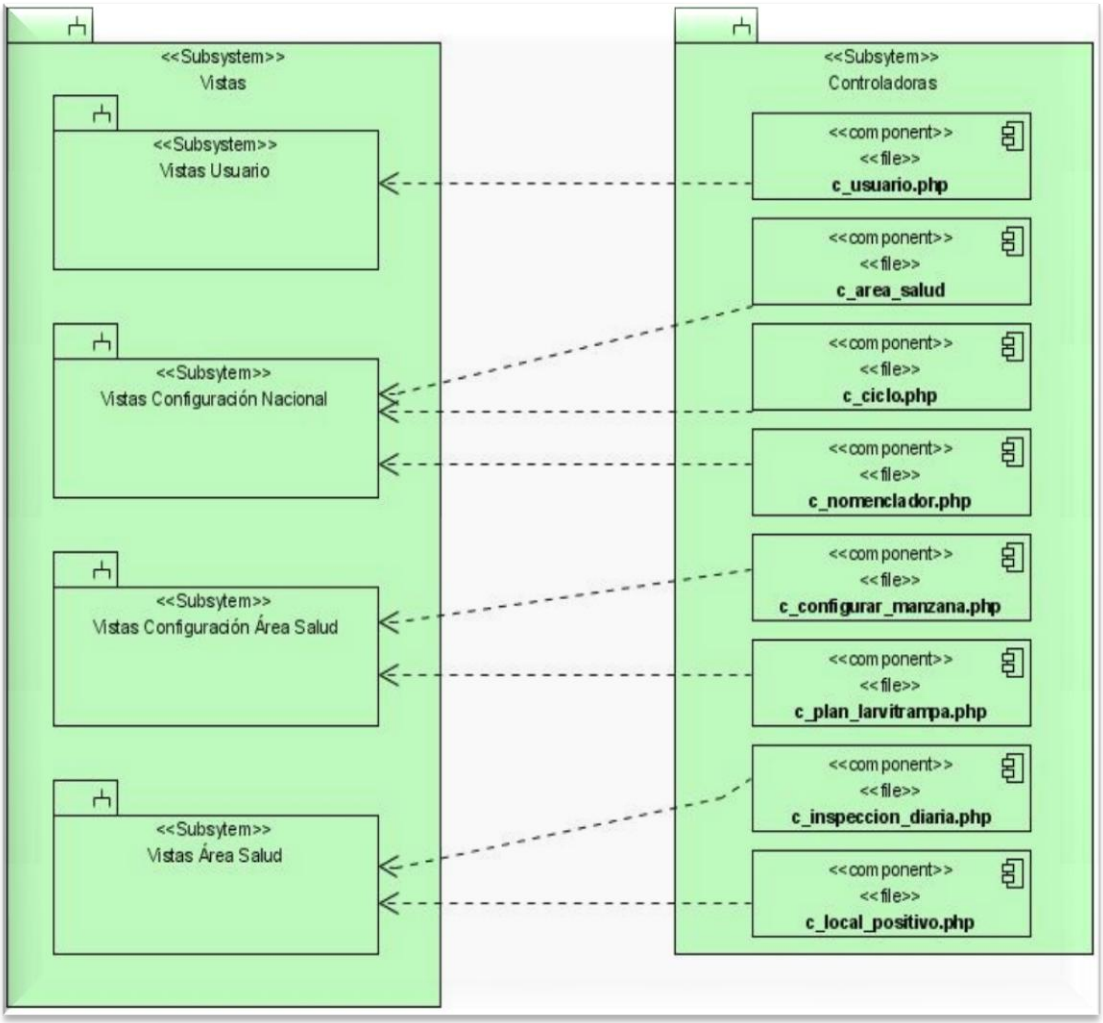

Figura 12. Relación entre el Subsistema Vistas y el Subsistema Controladoras.

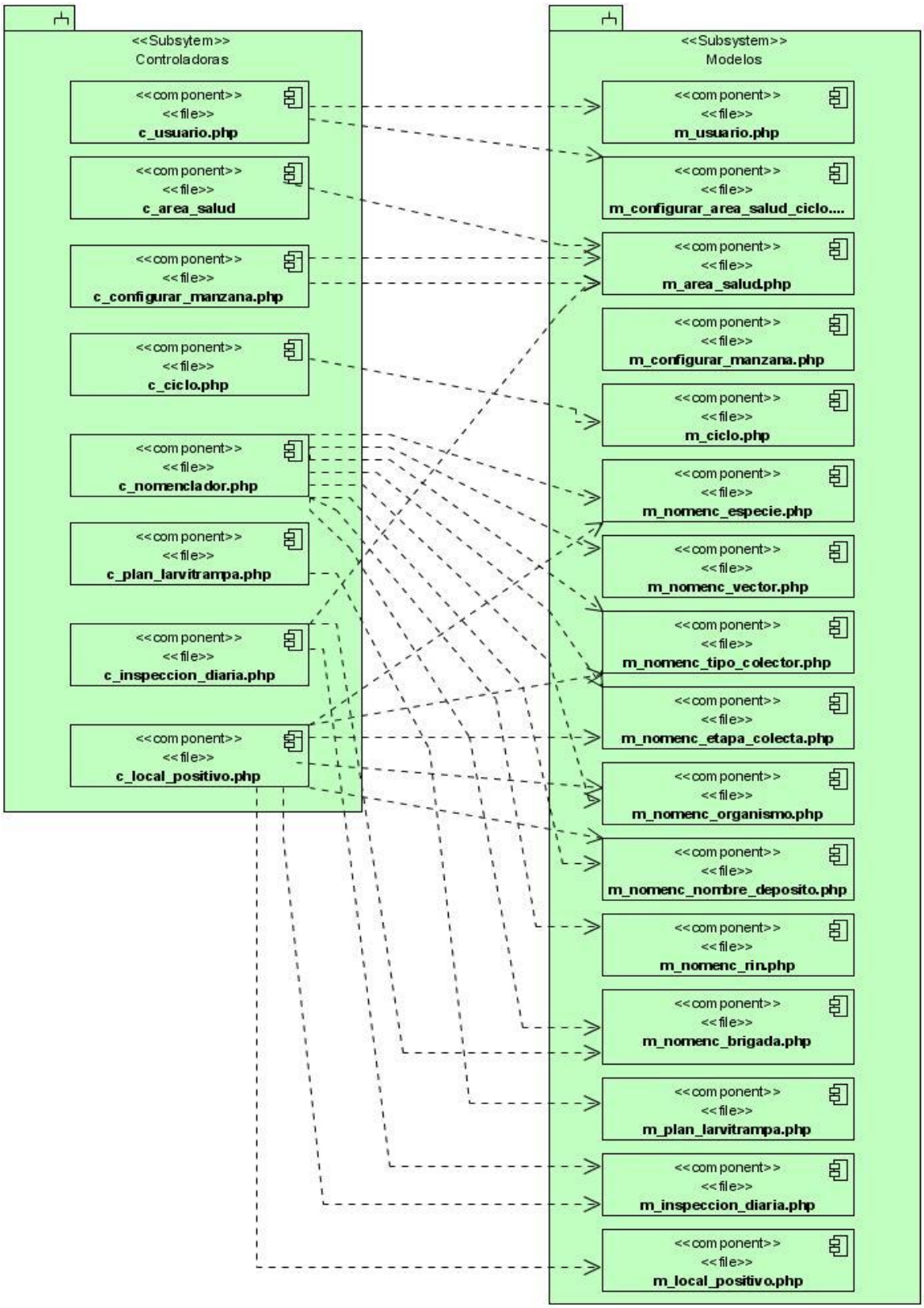

Figura 13. Relación entre el Subsistema Controladoras y el Subsistema Modelos.

Conclusiones

En este capítulo se hizo un análisis de los elementos de arquitectura presentes en el desarrollo del software; definiéndose como centralizada, en los servidores de INFOMED. Está basada en componentes, en tres capas, y en el patrón Modelo-Vista-Controlador. Además se definió la configuración de hardware en la cual se desplegará el sistema y se mostraron los subsistemas de implementación en los que se dividió la aplicación.

Capítulo 3: Descripción y Análisis de la solución propuesta

Introducción

El capítulo que a continuación se presenta, aborda primeramente las estrategias a seguir para la integración con otros módulos, además de la descripción de las clases del sistema y de las tablas de la Base de Datos; observándose también el Diagrama Entidad Relación de la misma.

3.1 Integración con otros Módulos o Sistemas

La utilización de una arquitectura basada en componentes, donde cada módulo es un componente con funcionalidad propia y que utiliza los servicios brindados por otros componentes hacen que el funcionamiento del Sistema de Vectores dependa de otros componentes del RIS, garantizando la reutilización de los datos y evitando la duplicidad de la información.

Los componentes o módulos del RIS que fueron identificados como necesarios para cumplimentar las funcionalidades que el sistema brinda son: el Sistema de Autenticación, Autorización y Auditoría (SAAA), el Registro de Unidades de Salud (RUS), el Registro de Localidades (RL) y el Registro de Ubicación (RU) .

3.1.1 Componente de Seguridad (SAAA)

Este componente se utiliza en la aplicación de vectores para llevar a cabo la autenticación del usuario en el sistema, la cual consiste en suministrar un nombre de usuario único y una contraseña. Si el usuario no se encuentra registrado se reportará un error de acceso. Si el usuario autenticado se encuentra registrado se autoriza su acceso y se crea un certificado digital que contiene un identificador único (token) de 32 caracteres, que tiene como información: el identificador del usuario, el nivel al que pertenece (Nacional, Provincial, Municipal o Unidad de Salud) y el identificador de nivel de acceso.

3.1.2 Registro de Unidades de Salud (RUS)

Este registro tiene almacenada la información correspondiente a las unidades de salud de Cuba. Una de las funcionalidades del RUS es la de relacionar un tipo de unidad de salud con sus atributos (cualitativos o de capacidad), esta relación incluye también la combinación de estos atributos con las unidades de salud que los posean, a los que se le puede asignar un valor que indique si son o no un área de salud.

Uno de los servicios web disponibles en el RUS y que se utiliza en Vectores, es el que permite la búsqueda de las áreas de salud. La información que ofrece este servicio se realiza teniendo en cuenta un conjunto de parámetros de entrada, como el nombre, la ubicación geográfica (Provincia o Municipio) o los atributos cualitativos o de capacidad de las unidades de salud.

El sistema de Vectores se comunica con el RUS para obtener dadas una provincia y un municipio las áreas de salud que pertenecen a dicha provincia y dicho municipio.

3.1.3 Registro de Ubicación (RU)

El RU es uno de los componentes no médicos del Registro Informatizado de Salud. El mismo gestiona la información de las Provincias, los Municipios, Localidades, Calles y Manzanas del país. Los servicios que ofrece este componente se utilizan en la aplicación de Vectores para introducir las direcciones de los diferentes tipos de locales: viviendas, centros de trabajo y terrenos baldíos que constituyan locales donde exista un foco de mosquito *Aedes Aegypti*.

3.1.4 Registro de Localidades (RL)

El Registro de Localidades de forma centralizada gestiona la información de los Consejos Populares, Circunscripciones, Zonas CDR y CDR de cada municipio del país. Los servicios que este registro ofrece se utilizan para introducir las direcciones de los diferentes tipos de locales: viviendas, centros de trabajo y terrenos baldíos que constituyan locales donde exista un foco de mosquito *Aedes Aegypti*.

3.2 Descripción de las clases u operaciones necesarias

A continuación se presentan las descripciones de las clases controladoras y modelos implementadas para llevar a cabo el desarrollo de la aplicación.

3.2.1 Clases Controladoras

Nombre C_Usuario	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	C_Usuario()
Descripción:	Constructor de la clase
Nombre:	index()

Descripción:	Método que se ejecuta por defecto y si el usuario no está autenticado lo envía a la pantalla de autenticación.
Nombre:	No_Acces()
Descripción:	Método que remite a la página de error de acceso.
Nombre:	AutenticateSAAA()
Descripción:	Método que valida la autenticación de los distintos usuarios
Nombre:	Mostrar_Pagina_Principal()
Descripción:	Método que muestra la página principal.
Nombre:	Salir()
Descripción:	Método que permite la salida del sistema del usuario que esté autenticado en ese momento.
Nombre:	Asignar_Ciclo($id_unidad_salud)
Descripción:	Método que según el usuario que sea lo ubica en su área de salud y le asigna un ciclo.
Nombre:	Asignar_Ciclo_2()
Descripción:	Método que en caso de que el usuario tenga posibilidad de entrar a más de un área de salud le da a escoger con cuál quiere trabajar.
Nombre:	Cargar_Combo_Areas_Salud($areas_salud)
Descripción:	Método que carga las diferentes divisiones por ciclo de un área de salud.

Tabla 1. Descripción de la clase controladora C_Usuario.

Nombre: C_Ciclo	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	C_Ciclo()
Descripción:	Constructor de la clase.
Nombre:	Index()
Descripción:	Método que se ejecuta cada vez que la clase es llamada.
Nombre:	Cargar_Combo_Tipo_Ciclo()
Descripción:	Método que si se ha escogido un ciclo en el combo box envía este valor al método Visualizar_Por_Tipo_Ciclo() y en caso de que no se haya seleccionado ningún ciclo muestra la misma página.

Nombre:	Visualizar_Por_Tipo_Ciclo($id_tipo_ciclo)
Descripción:	Método en el que según el tipo de ciclo pasado por parámetro carga los valores y construye la página de presentación correspondiente.
Nombre:	Insertar()
Descripción:	Método que se encarga de insertar un ciclo.
Nombre:	Eliminar()
Descripción:	Método que se encarga de eliminar los ciclos no deseados.
Nombre:	Diciembre()
Descripción:	Método que verifica si la fecha actual es diciembre para permitir la entrada de los nuevos ciclos.

Tabla 2. Descripción de la clase controladora C_Ciclo.

Nombre: C_Area_Salud	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	C_Area_Salud()
Descripción:	Constructor de la clase.
Nombre:	index()
Descripción:	Método que se ejecuta con solo llamar a la clase, se encarga de mandar a cargar el combo box de las provincias y la página de presentación correspondiente.
Nombre:	Municipios($id_prov='')
Descripción:	Método que dada una provincia, carga los municipios correspondientes.
Nombre:	Provincias()
Descripción:	Método que carga las provincias del país.
Nombre:	Cargar_combo_provincias()
Descripción:	Método que pasa para un combo box todas las provincias del país.
Nombre:	Listar_Areas_Salud($id_provincia,$id_municipio)
Descripción:	Método que dada una provincia y un municipio devuelve las áreas de salud correspondientes.
Nombre:	Visualizar()
Descripción:	Método encargado de construir la página que muestra las áreas de salud para

	ser configuradas.
Nombre:	Insertar()
Descripción:	Método que se encarga de insertar las áreas de salud escogidas en la base de datos.

Tabla 3. Descripción de la clase controladora C_Area_Salud.

Nombre: C_Inspeccion_Diaria	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	C_Inspeccion_Diaria()
Descripción:	Constructor de la clase.
Nombre:	Index()
Descripción:	Método que se ejecuta con solo llamar a la clase y que carga el ciclo actual en que se encuentra el área de salud.
Nombre:	Listar_Localidades($id_provincia,$id_municipio)
Descripción:	Método que carga las localidades dada una provincia y un municipio del RU.
Nombre:	Listar_Manzanas_Por_Localidad($id_localidad)
Descripción:	Método que carga las manzanas dada una localidad del RU.
Nombre:	Cargar_Combo_Localidades()
Descripción:	Método que construye un combo con las localidades de la provincia y el municipio al que pertenece el área de salud.
Nombre:	Visualizar($fecha_inspeccion)
Descripción:	Método que muestra las inspecciones realizadas dada una fecha.
Nombre:	Insertar()
Descripción:	Método encargado de insertar una inspección.
Nombre:	Eliminar()
Descripción:	Método encargado de eliminar una inspección determinada..
Nombre:	Actualizar()
Descripción:	Método encargado de actualizar una inspección determinada.

Tabla 4. Descripción de la clase controladora C_Inspeccion_Diaria.

Nombre: C_Local_Positivo

Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	C_Local_Positivo()
Descripción:	Constructor de la clase.
Nombre:	Index()
Descripción:	Método que se ejecuta con solo llamar a la clase.
Nombre:	Visualizar_Locales_Positivos()
Descripción:	Método que visualiza los locales positivos correspondientes a una inspección diaria..
Nombre:	Nuevo_Foco()
Descripción:	Método que carga los datos necesarios para visualizar la página de insertar nuevos locales positivos.
Nombre:	Listar_Consejos_Populares_Por_Localidad($id_localidad)
Descripción:	Método que obtiene los consejos populares dada una localidad del RL
Nombre:	Buscar_Consejo_Popular($id_consejo_popular)
Descripción:	Método que obtiene el nombre de un consejo popular dado del RL.
Nombre:	Listar_Circunscripciones_Por_Consejo_Popular($id_consejo_popular)
Descripción:	Método que obtiene las circunscripciones dado un consejo popular del RL
Nombre:	Buscar_Circunscripcion($id_consejo_popular, $id_circunscripcion)
Descripción:	Método que obtiene el nombre de una circunscripción dada del RL.
Nombre:	Listar_Calles_Por_Manzana($id_manzana)
Descripción:	Método que obtiene las calles dada una manzana del RU.
Nombre:	Buscar_Calle($id_calle)
Descripción:	Método que obtiene el nombre de una calle determinada.
Nombre:	Cargar_Combo_Consejos_Populares($id_consejo_popular)
Descripción:	Método que devuelve los consejos populares en un combobox.
Nombre:	Cargar_Combo_Calles($id_calle = -1)
Descripción:	Método que devuelve las calles en formato de dropdownlist.
Nombre:	Listar_Tipo_Colectores_Por_Etapa_Colecta($id_etapa_colecta)
Descripción:	Método que devuelve los diferentes tipos de colectores dada una etapa de colecta en formato de dropdownlist.
Nombre:	Listar_Depositos_Por_Tipo($tipo_deposito)
Descripción:	Método que devuelve los depósitos dado un tipo de depósito en formato de

	dropdownlist.
Nombre:	Visualizar_Depositos()
Descripción:	Método que visualiza los depósitos que han sido insertados durante la entrada de un nuevo foco.
Nombre:	Insertar_Deposito()
Descripción:	Método que asigna un depósito a un foco.
Nombre:	Eliminar_Deposito()
Descripción:	Método que elimina un depósito del arreglo de depósitos asignados a un foco.
Nombre:	Obtener_Deposito($id_deposito)
Descripción:	Método para obtener el nombre de un depósito determinado.
Nombre:	Visualizar_Especies_Asociadas($id_deposito="")
Descripción:	Método que permite visualizar las especies asociadas a un depósito determinado.
Nombre:	Insertar_Especie_Asociada()
Descripción:	Método que asigna una especie asociada a un depósito.
Nombre:	Eliminar_Especie_Asociada()
Descripción:	Método que elimina una especie asociada del arreglo de especies asociadas a un depósito.
Nombre:	Obtener_Especies_Asociadas_Deposito($id_deposito)
Descripción:	Método para obtener las especies asociadas a un depósito.
Nombre:	Obtener_Rango($arreglo, $offset)
Descripción:	Método para obtener un determinado rango de elementos dado un arreglo.
Nombre:	Insertar_Local_Positivo()
Descripción:	Método que inserta un local positivo con sus depósitos y las especies asociadas a los mismos.
Nombre:	Eliminar_Local_Positivo()
Descripción:	Método para eliminar un local positivo determinado.
Nombre:	Actualizar_Local_Positivo()
Descripción:	Método para actualizar un local positivo.
Nombre:	Insertar_Deposito_2()
Descripción:	Método que permite asignar un depósito a un local cuando este está siendo actualizado.
Nombre:	Eliminar_Deposito_2()

Descripción:	Método que permite eliminar un depósito del arreglo de depósitos asignados a un local cuando este está siendo actualizado.
Nombre:	Visualizar_Especies_Asociadas_2($posicion_deposito)
Descripción:	Método que muestra las especies asociadas a un depósito cuando el mismo está siendo actualizado.
Nombre:	Insertar_Especie_Asociada_2()
Descripción:	Método que permite asignar una especie asociada a un depósito cuando se está actualizando el mismo.
Nombre:	Eliminar_Especie_Asociada_2()
Descripción:	Método que permite eliminar una especie asociada a un depósito cuando el local positivo está siendo actualizado.
Nombre:	Crear_Opciones_Circunscripciones($estrucsalida,$id)
Descripción:	Método que construye un combo box con los datos de las circunscripciones.
Nombre:	Crear_Opciones_Tipo_Colectores($estrucsalida,$id)
Descripción:	Método que construye un combo box con los tipos de colectores.
Nombre:	Ver_Detalle_Local_Positivo($especies_asociadas_a = array())
Descripción:	Método que se encarga de visualizar los datos de un local positivo determinado.
Nombre:	Ver_Especies_Asociadas_Detalle($id_deposito)
Descripción:	Método que muestra las especies asociadas a un depósito cuando se están mostrando los detalles del mismo

Tabla 5. Descripción de la clase controladora C_Local_Positivo.

Nombre: C_Configurar_Manzana	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	C_Configurar_Manzana()
Descripción:	Constructor de la clase
Nombre:	index()
Descripción:	Primer método que se ejecuta al cargar la clase.
Nombre:	Listar_Localidades($id_municipio ,$id_provincia
Descripción:	Método que dada una provincia y un municipio lista las localidades

Nombre:	Listar_Manzanas_Por_Localidad($id_localidad)
Descripción:	Método que dada una localidad devuelve las manzanas correspondientes
Nombre:	Cargar_Combo_Localidades()
Descripción:	Método que carga el combo box de las localidades.
Nombre:	Listar_Localidad_Por_ID($id_localidad)
Descripción:	Método que dado un id de localidad lista los datos pertenecientes a la misma.
Nombre:	Listar_Manzana_Por_ID($id_localidad ,$id_manzana)
Descripción:	Método que dado un id de localidad y uno de manzana lista los datos pertenecientes a la misma.
Nombre:	Visualizar($id_area_salud)
Descripción:	Método encargado de construir la página con los datos correspondientes.
Nombre:	Insertar()
Descripción:	Método encargado de insertar una manzana.
Nombre:	Actualizar()
Descripción:	Método encargado de actualizar los datos de una manzana.
Nombre:	Eliminar()
Descripción:	Método encargado de eliminar una manzana.

Tabla 6. Descripción de la clase controladora C_Configurar_Manzana.

Nombre: C_Plan_Larvitrampa	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	C_Plan_Larvitrampa()
Descripción:	Constructor de la clase
Nombre:	index()
Descripción:	Método que se ejecuta por defecto al llamar a la clase.
Nombre:	Visualizar_Plan()
Descripción:	Método que visualiza los planes de larvitrampas que hay hasta el momento en esa área de salud. Gestiona todos los datos para esta operación y carga y construye la vista.
Nombre:	Insertar()
Descripción:	Método encargado de insertar un nuevo plan de larvitrampas.

Nombre:	Eliminar()
Descripción:	Método encargado de eliminar un plan de larvitrampas.
Nombre:	Actualizar()
Descripción:	Método encargado de actualizar un plan de larvitrampas.

Tabla 7. Descripción de la clase controladora C_Plan_Larvitrampa.

Nombre: C_Nomenclador	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	C_Nomenclador()
Descripción:	Constructor de la clase.
Nombre:	index()
Descripción:	Método que construye la vista con la que se va a trabajar y gestiona los datos del nomenclador seleccionado.
Nombre:	Index_Brigada()
Descripción:	Método que gestiona el id del área de salud y se lo manda al método index para que lo incluya en los datos de la brigada al cargar la vista.
Nombre:	Gestionar_Tabla($tabla,$view)
Descripción:	Método que se encarga de buscar los datos del nomenclador seleccionado para construir la vista correspondiente.
Nombre:	Insertar()
Descripción:	Método que se encarga de insertar los datos de un nomenclador.
Nombre:	Eliminar()
Descripción:	Método que se encarga de eliminar un nomenclador dado.
Nombre:	Actualizar()
Descripción:	Método que se encarga de actualizar los datos de un nomenclador dado.

Tabla 8. Descripción de la clase controladora C_Nomenclador.

3.2.2 Clases Modelos

Nombre: M_Usuario	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	M_Usuario()
Descripción:	Constructor de la clase.
Nombre:	Get_Roles_4User($username)
Descripción:	Método que devuelve el rol dado un usuario.
Nombre:	Is_User_In_Rol($username,$rol)
Descripción:	Método que dado un usuario y un rol verifica si ese usuario tiene ese rol.
Nombre:	Get_Links_4User($username)
Descripción:	Método que recolecta los links a los que el usuario tiene acceso.
Nombre:	Have_Autorization($username,$link)
Descripción:	Método que recolecta los links a los que el usuario tiene acceso y se muestran en el menú de navegación.

Tabla 9. Descripción de la clase modelo M_Usuario.

Nombre: M_Configurar_Area_Salud_Ciclo	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	M_Configurar_Area_Salud_Ciclo()
Descripción:	Constructor de la clase.
Nombre:	Buscar_Areas_Por_Id_Unidad_Salud($id_unidad_salud)
Descripción:	Método que dado un id de unidad de salud busca si está configurada o no en ciclo y de no estarlo, la ubica.
Nombre:	Buscar_Areas_Por_Id_Area_Salud($id_area_salud)
Descripción:	Método que dado un id de área de salud busca si está configurada o no en ciclo y de no estarlo, la ubica.
Nombre:	Ubicar_Ciclo($id_area_salud)
Descripción:	Método que ubica una cierta área de salud mediante su id (dependiendo de si es de dos semanas, mensual o bimestral) en el ciclo en que le corresponde

	estar.

Tabla 10. Descripción de la clase modelo M_Configurar_Area_Salud_Ciclo.

Nombre M_Ciclo	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	M_Ciclo()
Descripción:	Constructor de la clase
Nombre:	Nombre_Ciclo($ciclo)
Descripción:	Método que determina el nombre del ciclo.
Nombre:	Insertar($ciclo)
Descripción:	Método que se encarga de insertar un ciclo en la base de datos.
Nombre:	Eliminar($id)
Descripción:	Método que se encarga de eliminar un ciclo.
Nombre:	Listar_Ciclos_Por_Tipo($tipo_ciclo,$per_page, $offset)
Descripción:	Método que devuelve los ciclos según el tipo especificado.
Nombre:	Validar_Ciclos_Completados()
Descripción:	Método que se encarga de validar los ciclos que ya se han completado, dando lugar a los ciclos nuevos.

Tabla 11. Descripción de la clase modelo M_Ciclo.

Nombre M_Area_Salud	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	M_Area_Salud()
Descripción:	Constructor de la clase.
Nombre:	Insertar_Area_Salud($areas_salud,$cantidad)
Descripción:	Método que se encarga de insertar los datos de las áreas de salud en la base de datos.
Nombre:	Buscar_Areas_Configuradas($provincia,$municipios)
Descripción:	Método que se encarga de buscar las áreas de salud que han sido configuradas.

Nombre:	Obtener_Ciclo_Actual($id_area_salud)
Descripción:	Método para obtener el ciclo actual de un área de salud.
Nombre:	Obtener_Prov_Munc_AS($id_area_salud)
Descripción:	Método para obtener la provincia y el municipio de un área de salud.

Tabla 12. Descripción de la clase modelo M_Area_Salud.

Nombre M_Inspeccion_Diaria	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	M_Inspeccion_Diaria()
Descripción:	Constructor de la clase.
Nombre:	Insertar($inspeccion_diaria)
Descripción:	Método que se encarga de insertar los datos de una inspección en la base de datos.
Nombre:	Eliminar($id)
Descripción:	Método que se encarga de eliminar una inspección determinada.
Nombre:	Actualizar($inspeccion_diaria)
Descripción:	Método que se encarga de actualizar los datos de una inspección determinada.
Nombre:	Calcular_Acumulado($inspeccion)
Descripción:	Método que se encarga de calcular los acumulados de los locales inspeccionados, locales cerrados, locales recuperados, locales inspeccionados, depósitos inspeccionados, destruidos, tratados y flameados hasta ese momento de una brigada en ese ciclo.
Nombre:	Inspeccion_Diaria($id)
Descripción:	Método que devuleve los datos de una inspección determinada.
Nombre:	Listar_Inspeccion_Diaria_Brigadas($query)
Descripción:	Método que devuelve los nombres de las brigadas de una lista de inspecciones.
Nombre:	Listar_Inspeccion_Diaria_Localidades($query)
Descripción:	Método que devuelve los nombres de las localidades de una lista de inspecciones.

Nombre:	Listar_Inspeccion_Diaria_Manzanas($query)
Descripción:	Método que devuelve los nombres de las manzanas de una lista de inspecciones.

Tabla 13. Descripción de la clase modelo M_Inspección_Diaria.

Nombre M_Local_Positivo	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	M_Local_Positivo()
Descripción:	Constructor de la clase.
Nombre:	Insertar_Casa($local_positivo,$casa,$depositos,$especies_asociadas,$circunscripcion,$consejo_popular,$calle)
Descripción:	Método que se encarga de insertar los datos de un local positivo en caso de que sea una casa.
Nombre:	Actualizar_Casa($local_positivo,$casa,$depositos,$especies_asociadas,$circunscripcion,$consejo_popular,$calle)
Descripción:	Método que se encarga de actualizar los datos de un local positivo en caso de que sea una casa.
Nombre:	Insertar_Edificio($local_positivo,$edificio,$depositos,$especies_asociadas,$circunscripcion,$consejo_popular,$calle)
Descripción:	Método que se encarga de insertar los datos de un local positivo en caso de que sea un edificio.
Nombre:	Actualizar_Edificio($local_positivo,$edificio,$depositos,$especies_asociadas,$circunscripcion,$consejo_popular,$calle)
Descripción:	Método que se encarga de actualizar los datos de un local positivo en caso de que sea un edificio.
Nombre:	Insertar_Terreno_Baldio($local_positivo,$depositos,$especies_asociadas,$circunscripcion,$consejo_popular,$calle)
Descripción:	Método que se encarga de insertar los datos de un local positivo en caso de que sea un terreno baldío
Nombre:	Actualizar_Terreno_Baldio($local_positivo,$depositos,$especies_asociadas,$circunscripcion,$consejo_popular,$calle)

Descripción:	Método que se encarga de actualizar los datos de un local positivo en caso de que sea un terreno baldío.
Nombre:	Insertar_Centro_Trabajo($local_positivo,$centro_trabajo,$depositos,$especies_asociadas,$circunscripcion,$consejo_popular,$calle)
Descripción:	Método que se encarga de insertar los datos de un local positivo en caso de que sea un centro de trabajo.
Nombre:	Actualizar_Centro_Trabajo($local_positivo,$centro_trabajo,$depositos,$especies_asociadas,$circunscripcion,$consejo_popular,$calle)
Descripción:	Método que se encarga de actualizar los datos de un local positivo en caso de que sea un centro de trabajo.
Nombre:	Insertar_Consejo_Popular($consejo_popular)
Descripción:	Método que se encarga de insertar los datos de un consejo popular.
Nombre:	Insertar_Circunscripcion($circunscripcion)
Descripción:	Método que se encarga de insertar los datos de una circunscripción.
Nombre:	Insertar_Calle($calle)
Descripción:	Método que se encarga de insertar los datos de una calle.
Nombre:	Insertar_Depositos($id_local_positivo, $depositos, $especies_asociadas)
Descripción:	Método que se encarga de insertar los depósitos correspondientes a un local positivo.
Nombre:	Actualizar_Depositos($id_local_positivo, $depositos, $especies_asociadas)
Descripción:	Método que se encarga de actualizar los depósitos correspondientes a un local positivo.
Nombre:	Insertar_Especies_Asociadas($id_deposito_ficticio,$id_deposito, $id_local_positivo, $especies_asociadas)
Descripción:	Método que se encarga de insertar las especies asociadas a un depósito.
Nombre:	Actualizar_Especies_Asociadas($id_deposito_ficticio,$id_deposito, $id_local_positivo, $especies_asociadas)
Descripción:	Método que se encarga de actualizar las especies asociadas a un depósito.
Nombre:	Eliminar_Local_Positivo($id)
Descripción:	Método que se encarga de eliminar un local positivo.
Nombre:	Listar_Local_Positivo($id_inspeccion_diaria,$per_page, $offset)
Descripción:	Método que lista los locales positivos de tipo terreno baldío correspondientes a una inspección.

Nombre:	Listar_Local_Positivo_Casa($id_inspeccion_diaria)
Descripción:	Método que lista los locales positivos de tipo casa correspondientes a una inspección.
Nombre:	Listar_Local_Positivo_Edificio($id_inspeccion_diaria)
Descripción:	Método que lista los locales positivos de tipo edificio correspondientes a una inspección.
Nombre:	Listar_Local_Positivo_Centro_Trabajo($id_inspeccion_diaria)
Descripción:	Método que lista los locales positivos de tipo centro de trabajo correspondientes a una inspección.
Nombre:	Listar_Local_Positivo_Calles($query)
Descripción:	Método que lista el nombre de las calles de un conjunto de locales positivos.
Nombre:	Local_Positivo($id)
Descripción:	Método que devuelve los datos de un local positivo determinado.
Nombre:	Listar_Depositos($id_local_positivo,$per_page, $offset)
Descripción:	Método que lista los depósitos pertenecientes a un local positivo determinado.
Nombre:	Listar_Especies_Asociadas_A_Deposito($id_deposito,$per_page, $offset)
Descripción:	Método que lista las especies asociadas a un depósito determinado.
Nombre:	Nombre_Consejo_Popular($id)
Descripción:	Método que devuelve el nombre de un consejo popular.
Nombre:	Nombre_Circunscripcion($id)
Descripción:	Método que devuelve el nombre de una circunscripción determinada.
Nombre:	Nombre_Calle($id)
Descripción:	Método que devuelve el nombre de una calle determinada.

Tabla 14. Descripción de la clase modelo M_Local_Positivo.

Nombre: M_Configurar_Manzana	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	M_Configurar_Manzana()
Descripción:	Constructor de la clase.
Nombre:	Insertar_Localidad_Manzana($localidad,$manzana)
Descripción:	Método que se encarga de insertar una manzana en la base de datos.

Nombre:	Listar_Manzanas_Por_Municipio_AS($per_page, $offset)
Descripción:	Método que devuelve las manzanas según el municipio.
Nombre:	Eliminar($id)
Descripción:	Método que se encarga de eliminar una manzana.
Nombre:	Actualizar($manzana)
Descripción:	Método que se encarga de actualziar los datos de una manzana.
Nombre:	Buscar_Manzana($id)
Descripción:	Método que devuelve los datos de una manzana.

Tabla 15. Descripción de la clase modelo M_Configurar_Manzana.

Nombre: M_Plan_Larvitrampa	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	M_Plan_Larvitrampa()
Descripción:	Constructor de la clase.
Nombre:	Insertar_Plan_Larvitrampa($plan_larv)
Descripción:	Método que se encarga de insertar un plan de larvitrampas en la base de datos.
Nombre:	Actualizar_Plan_Larvitrampa($plan_larv)
Descripción:	Método que se encarga de actualizar los datos de un plan de larvitrampas dado.
Nombre:	Insertar_Plan_Larvitrampa_Fin($plan_larv)
Descripción:	Método que se encarga de insertar los datos de un plan de larvitrampas después de concluido.
Nombre:	Eliminar($id)
Descripción:	Método que elimina un plan de larvitrampas dado.
Nombre:	Actualizar($plan_larvitrampa)
Descripción:	Método que se encarga de actualizar los datos de un plan de larvitrampas
Nombre:	Ultima_Gestion_Larvitrampa()
Descripción:	Método que devuelve la última gestión de larvitrampas realizada.
Nombre:	Listar_Plan_Larvitrampas($id_area_salud, $per_page, $offset)
Descripción:	Muestra los planes de larvitrampas.

Nombre:	Plan_Larvitrampa($id)
Descripción:	Método que se encarga de verificar si ha superado el número días para actualizar.

Tabla 16. Descripción de la clase modelo M_Plan_Larvitrampa.

Nombre: M_Nomenc_Brigada	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	M_Nomenc_Brigada ()
Descripción:	Constructor de la clase.
Nombre:	Insertar($nomenc)
Descripción:	Método que se encarga de insertar una brigada en la base de datos.
Nombre:	Eliminar($id)
Descripción:	Método que se encarga de eliminar una brigada.
Nombre:	Actualizar($nomenc)
Descripción:	Método que se encarga de actualizar los datos de una brigada.
Nombre:	Listar_Nomencladores($nomenc,$per_page, $offset)
Descripción:	Método que lista las brigadas existentes en la base de datos.
Nombre:	Nomenclador($nomenc,$id)
Descripción:	Método que devuelve los datos de una brigada determinada
Nombre:	CB_Brigadas($id_unidad_salud, $id_brigada = -1)
Decripción:	Método que carga las brigadas dada un área de salud en formato de dropdownlist.

Tabla 17. Descripción de la clase modelo M_Nomenc_Brigada.

Nombre: M_Nomenc_Especie()	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	M_Nomenc_Especie()
Descripción:	Constructor de la clase.
Nombre:	Insertar($nomenc)
Descripción:	Método que se encarga de insertar una especie en la base de datos

Nombre:	Eliminar($id)
Descripción:	Método que se encarga de eliminar una especie.
Nombre:	Actualizar($nomenc)
Descripción:	Método que se encarga de actualizar los datos de una especie.
Nombre:	Listar_Nomencladores($nomenc,$per_page, $offset)
Descripción:	Método que se encarga de listar las especies existentes en la base de datos.
Nombre:	Nomenclador($nomenc,$id)
Descripción:	Método que devuelve los datos de una especie determinada.
Nombre:	CB_Especies($id_especie = -1)
Descripción:	Método que carga las especies en formato de dropdownlist.

Tabla 18. Descripción de la clase modelo M_Nomenc_Especie.

Nombre: M_Nomenc_Etapa_Colecta()	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	M_Nomenc_Etapa_Colecta()
Descripción:	Constructor de la clase.
Nombre:	Insertar($nomenc)
Descripción:	Método que se encarga de insertar una etapa de colecta en la base de datos.
Nombre:	Eliminar($id)
Descripción:	Método que se encarga de eliminar una etapa de colecta.
Nombre:	Actualizar($nomenc)
Descripción:	Método que se encarga de actualizar los datos de una etapa de colecta.
Nombre:	Listar_Nomencladores($nomenc,$per_page, $offset)
Descripción:	Método que lista las etapas de colecta existentes en la base de datos.
Nombre:	Nomenclador($nomenc,$id)
Descripción:	Método que devuelve los datos de una etapa de colecta determinada.
Nombre:	CB_Etapa_Colecta($id_etapa_colecta = -1)
Descripción:	Método que carga las etapas de colecta en formato de dropdownlist.

Tabla 19. Descripción de la clase modelo M_Nomenc_Etapa_Colecta.

Nombre: M_Nomenc_Nombre_Deposito()	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	M_Nomenc_Nombre_Deposito()
Descripción:	Constructor de la clase.
Nombre:	Insertar($nomenc)
Descripción:	Método que se encarga de insertar un nombre de depósito en la base de datos.
Nombre:	Eliminar($id)
Descripción:	Método que se encarga de eliminar un nombre de depósito determinado.
Nombre:	Actualizar($nomenc)
Descripción:	Método que se encarga de actualizar un nombre de depósito determinado.
Nombre:	Listar_Nomencladores($nomenc,$per_page, $offset)
Descripción:	Método que lista los nombres de depósitos existentes en la base de datos.
Nombre:	Nomenclador($nomenc,$id)
Descripción:	Método que devuelve un nombre de depósito determinado.
Nombre:	CB_Nombre_Depositos($tipo_deposito)
Descripción:	Método que devuelve los nombres de los depósitos en formato de dropdownlist.

Tabla 20. Descripción de la clase modelo M_Nomenc_Nombre_Deposito.

Nombre: M_Nomenc_Organismo()	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	M_Nomenc_Organismo()
Descripción:	Constructor de la clase.
Nombre:	Insertar($nomenc)
Descripción:	Método que se encarga de insertar un organismo en la base de datos
Nombre:	Eliminar($id)
Descripción:	Método que se encarga de eliminar un organismo determinado.
Nombre:	Actualizar($nomenc)
Descripción:	Método que se encarga de actualizar los datos de un organismo determinado.
Nombre:	Listar_Nomencladores($nomenc,$per_page, $offset)
Descripción:	Método que lista los organismos existentes en la base de datos.

Nombre:	Nomenclador($nomenc,$id)
Descripción:	Método que devuelve los datos de un organismo determinado.
Nombre:	CB_Nombre_Organismos($id_organismo = -1)
Descripción:	Método que devuelve los organismo en formato de dropdownlist.

Tabla 21. Descripción de la clase modelo M_Nomenc_Deposito.

Capítulo 3: Descripción y Análisis de la solución propuesta

3.3 Diseño de la Base de Datos

3.3.1 Modelo Lógico

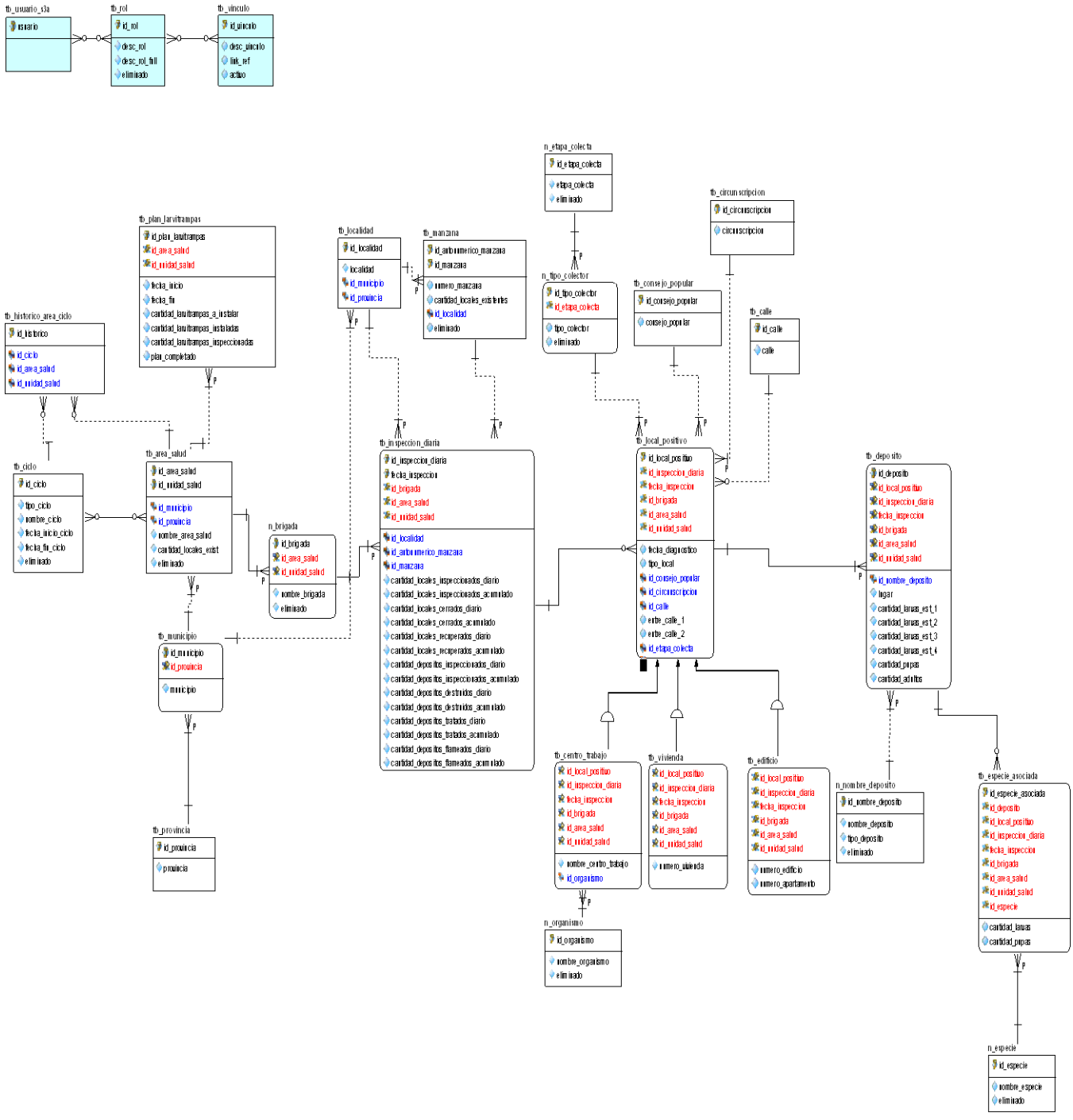

Figura 14 Modelo Lógico de la Base de Datos.

Capítulo 3: Descripción y Análisis de la solución propuesta

A continuación se muestra el modelo lógico fragmentado en partes para un mejor entendimiento.

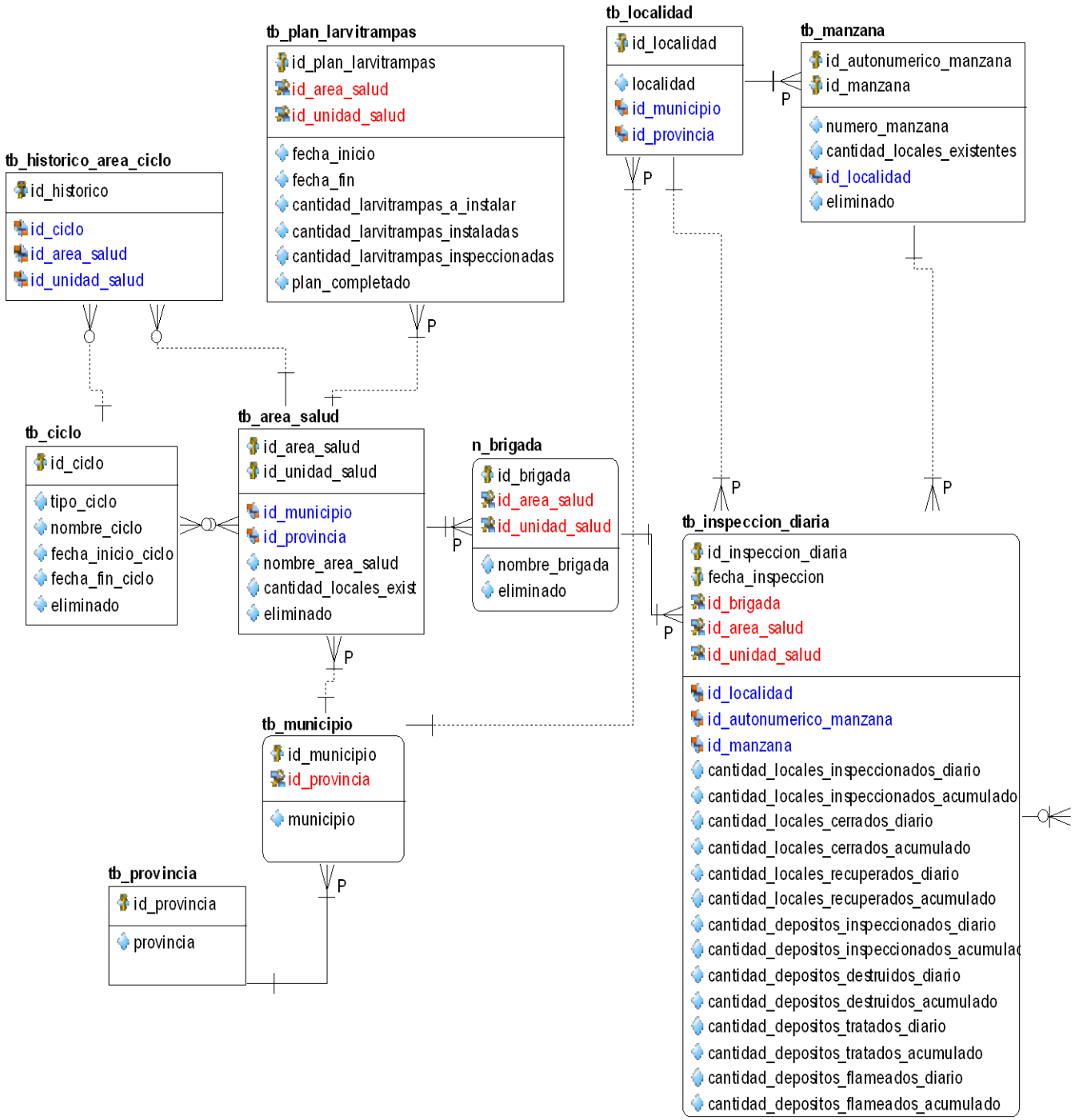

Figura 15. Modelo Lógico. (Fragmento 1)

Capítulo 3: Descripción y Análisis de la solución propuesta

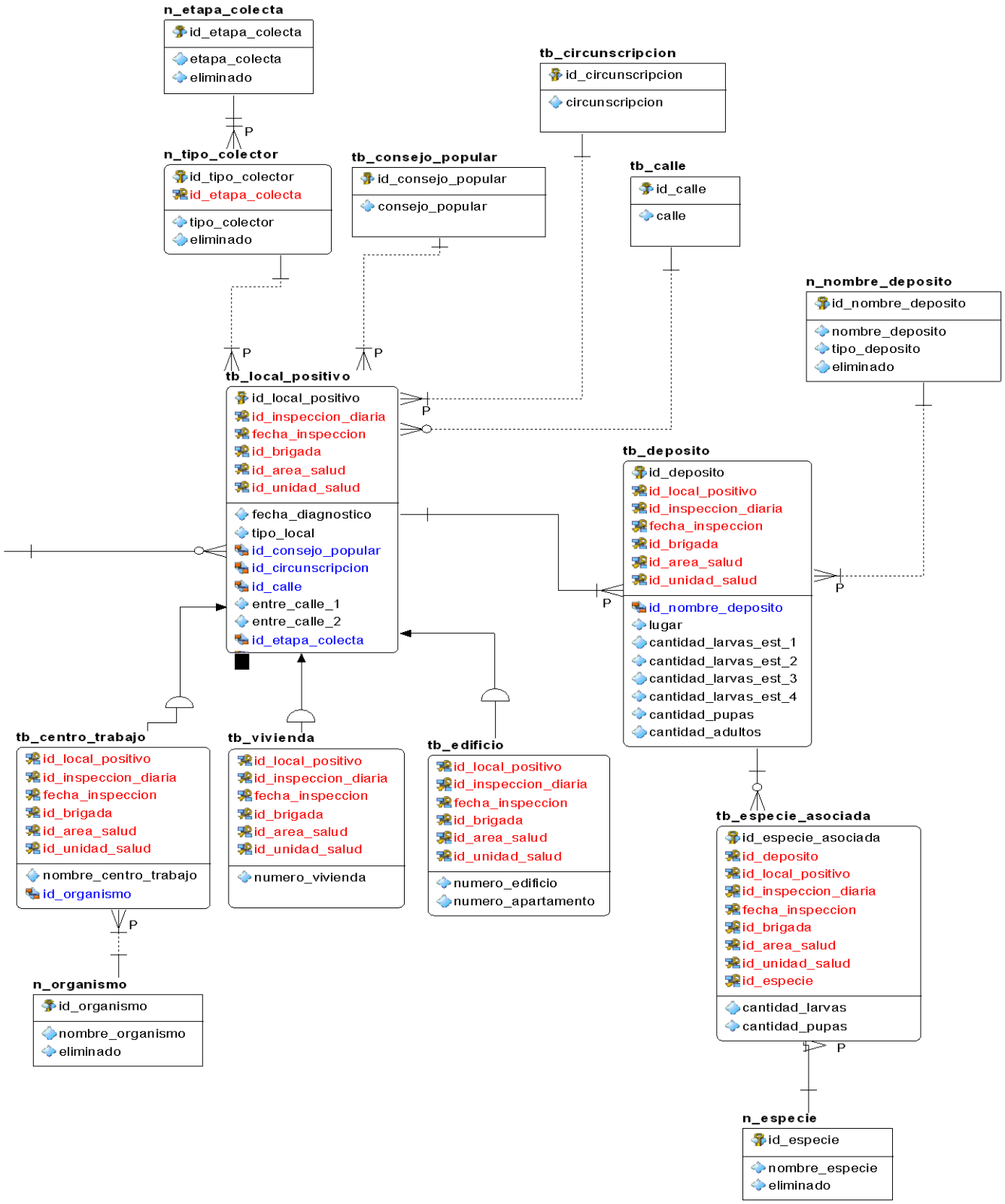

Figura 16. Modelo Lógico. (Fragmento 2)

Capítulo 3: Descripción y Análisis de la solución propuesta

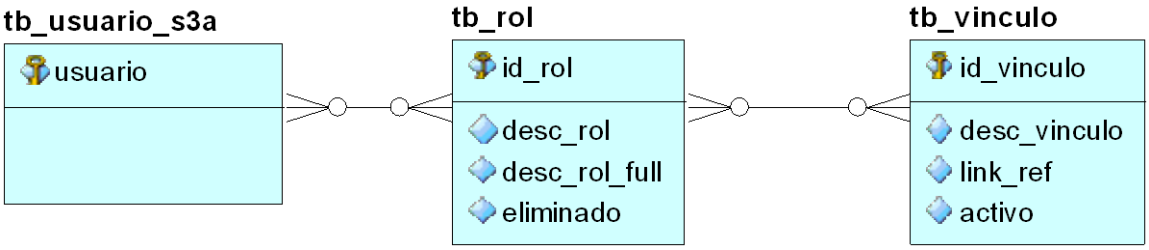

Figura 17. Modelo Lógico. (Fragmento 3)

3.3.2 Modelo Físico

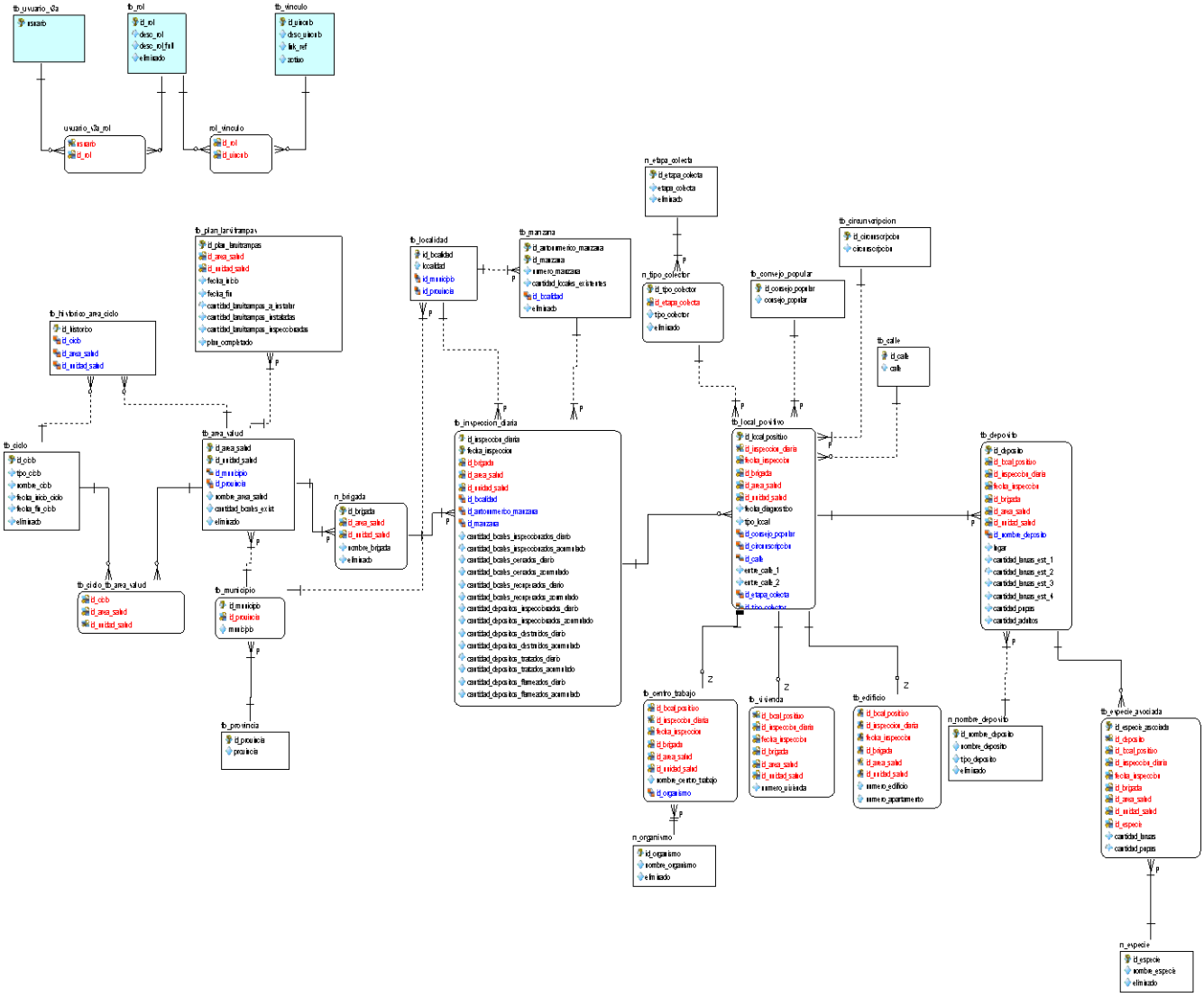

Figura 18. Modelo Físico de la Base de Datos.

Capítulo 3: Descripción y Análisis de la solución propuesta

A continuación se muestra el modelo físico fragmentado en partes para un mejor entendimiento.

Figura 19. Modelo Físico. (Fragmento1)

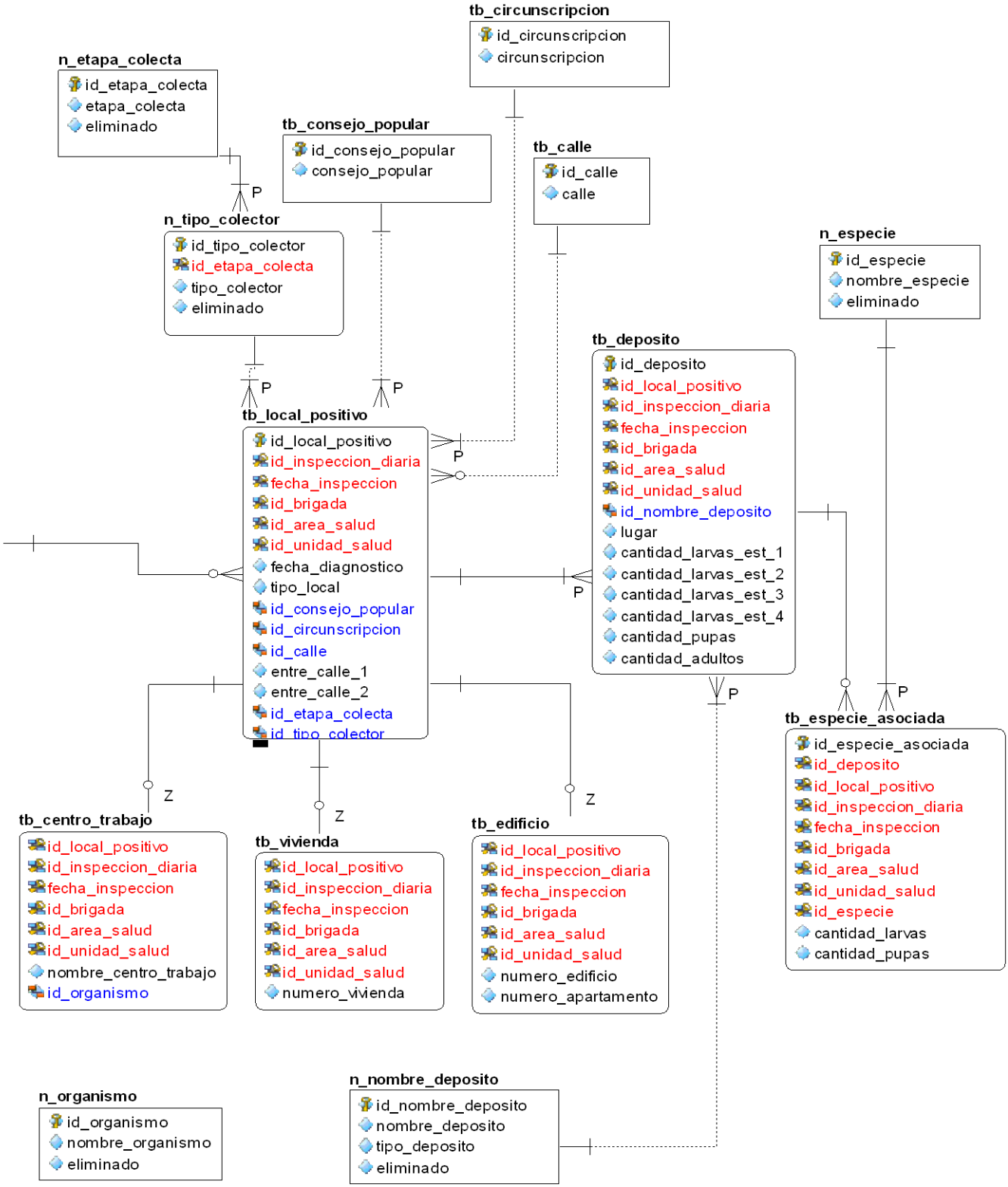

Figura 20. Modelo Físico (Fragmento 2) .

Capítulo 3: Descripción y Análisis de la solución propuesta

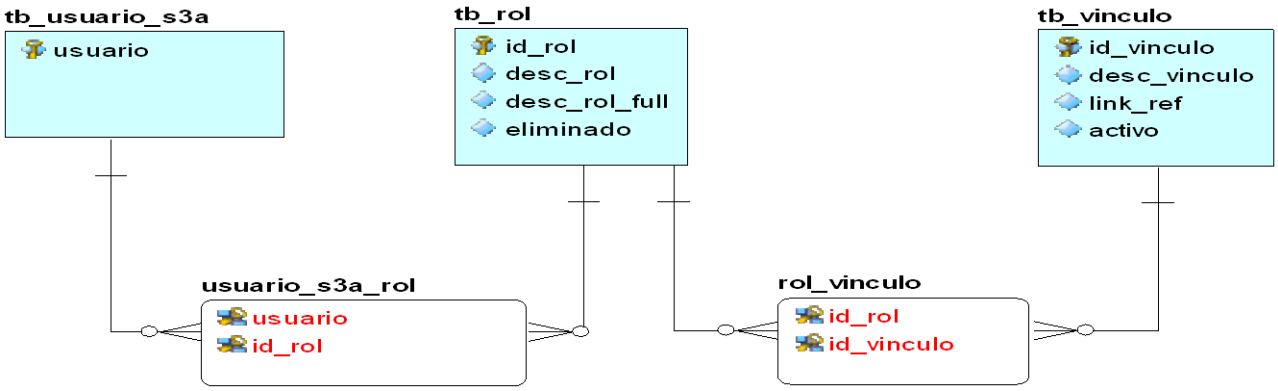

Figura 21. Modelo Físico. (Fragmento 3)

3.4 Descripción de las tablas

Nombre: n_brigada		
Descripción: Guarda los datos de las brigadas.		
Atributo	**Tipo**	**Descripción**
id_brigada	INTEGER	Forma parte de la llave primaria de la tabla.
id_area_salud	INTEGER	Forma parte de la llave primaria de la tabla. Es importada de la tabla tb_area_salud
id_unidad_salud	INTEGER	Forma parte de la llave primaria de la tabla. Es importada de la tabla tb_area_salud
nombre_brigada	VARCHAR	Guarda el nombre con el que se identifica la brigada
eliminado	TINYINT	Guarda un valor que puede ser 1 (se el elemento está eliminado), 0 (si el elemento está activo)

Tabla 22. Descripción de la tabla n_brigada.

Nombre: n_especie		
Descripción: Guarda los datos de las especies.		
Atributo	**Tipo**	**Descripción**
id_especie	INTEGER	Llave primaria de la tabla.
nombre_especie	VARCHAR	Guarda el nombre de la especie.
eliminado	TINYINT	Guarda un valor que puede ser 1 (se el elemento está eliminado), 0 (si el elemento está activo)

Tabla 23. Descripción de la tabla n_especie.

Nombre: n_etapa_colecta		
Descripción: Guarda las etapas de colecta		
Atributo	**Tipo**	**Descripción**
id_etapa_colecta	INTEGER	Llave primaria de la tabla.
etapa_colecta	VARCHAR	Guarda la etapa de colecta.
eliminado	TINYINT	Guarda un valor que puede ser 1 (se el elemento está eliminado), 0 (si el elemento está activo)

Tabla 24. Descripción de la tabla n_etapa_colecta.

Nombre: n_nombre_deposito		
Descripción: Guarda los nombres de los depositos		
Atributo	**Tipo**	**Descripción**
id_nombre_deposito	INTEGER	Llave primaria de la tabla.
nombre_deposito	VARCHAR	Guarda el nombre del depósito.
tipo_deposito	VARCHAR	Guarda el tipo de deposito (artificial o natural)
eliminado	TINYINT	Guarda un valor que puede ser 1 (se el elemento está eliminado), 0 (si el elemento está activo)

Tabla 25. Descripción de la tabla n_nombre_deposito.

Nombre: n_organismo		
Descripción: Guarda los nombres de los organismos		
Atributo	**Tipo**	**Descripción**
id_organismo	INTEGER	Llave primaria de la tabla.
nombre_organismo	VARCHAR	Guarda el nombre del organismo.
eliminado	TINYINT	Guarda un valor que puede ser 1 (se el elemento está eliminado), 0 (si el elemento está activo)

Tabla 26. Descripción de la tabla n_organismo.

Nombre: n_tipo_colector		
Descripción: Almacena los datos de los distintos tipos de colectores		
Atributo	**Tipo**	**Descripción**
id_tipo_colector	INTEGER	Forma parte de la llave primaria del la tabla.
id_etapa_colecta	INTEGER	Forma parte de la llave primaria del la tabla. Es importada de la tabla n_etapa_colecta
tipo_colector	VARCHAR	Guarda el nombre del tipo de colector
eliminado	TINYINT	Guarda un valor que puede ser 1 (se el elemento está eliminado), 0 (si el elemento está activo)

Tabla 27. Descripción de la tabla n_tipo_colector.

Nombre: rol_vinculo		
Descripción: Tabla que se crea por la unión de la tabla tb_rol y tb_vinculo. Almacena una especie de privilegios por rol.		
Atributo	**Tipo**	**Descripción**
id_rol	INTEGER	Forma parte de la llave primaria de la tabla. Importada de la tabla tb_rol
id_vinculo	INTEGER	Forma parte de la llave primaria de la tabla tb_vinculo.

Tabla 28. Descripción de la tabla rol_vinculo.

Nombre: tb_area_salud		
Descripción: Contiene los datos de las áreas de salud		
Atributo	**Tipo**	**Descripción**
id_area_salud	INTEGER	Llave primaria de la tabla, se crea auto numéricamente.
id_unidad_salud	INTEGER	Guarda el id de la unidad de salud
id_municipio	INTEGER	Guarda el id del municipio al que pertenece
Id_provincia	INTEGER	Guarda el id de la provincia a la que pertenece
nombre_area_salud	VARCHAR	Guarda el nombre del área de salud formado por el tipo de ciclo y el nombre de la unidad de salud
cantidad_locales_exist	INTEGER	Guarda la cantidad de locales existentes
eliminado	TINYINT	Guarda un valor para conocer su estado: 1 eliminado, 2 no eliminado

Tabla 29. Descripción de la tabla tb_area_salud.

Nombre: tb_calle		
Descripción: Contiene los datos de las calles		
Atributo	**Tipo**	**Descripción**
id_calle	INTEGER	Llave primaria de la tabla
calle	VARCHAR	Guarda el nombre o número de la calle

Tabla 30. Descripción de la tabla tb_calle.

Nombre: tb_centro_trabajo		
Descripción: Guarda los datos de los centros de trabajo		
Atributo	**Tipo**	**Descripción**
id_loca_positivo	INTEGER	Conforma la llave primaria de la tabla. Importada de tb_local_positivo

id_inspeccion_diaria	INTEGER	Conforma la llave primaria de la tabla. Importada de tb_inspecion_diaria
fecha_inspeccion	DATE	Conforma la llave primaria de la tabla. Importada de tb_inspecion_diaria
id_brigada	INTEGER	Conforma la llave primaria de la tabla. Importada de n_brigada
id_area_salud	INTEGER	Conforma la llave primaria de la tabla. Importada de tb_area_salud
id_unidad_salud	INTEGER	Conforma la llave primaria de la tabla. Importada de tb_area_salud
nombre_centro_trabajo	VARCHAR	Guarda el nombre del centro de trabajo
id_organismo	INTEGER	Guarda el id del organismo al que pertenece

Tabla 31. Descripción de la tabla tb_centro_trabajo.

Nombre: tb_ciclo		
Descripción: Guarda los datos de los ciclos		
Atributo	**Tipo**	**Descripción**
id_ciclo	INTEGER	Llave primaria de la tabla.
tipo_ciclo	INTEGER	Guarda el tipo de ciclo, que puede ser de dos semanas, mensual, bimestral.
nombre_ciclo	VARCHAR	Guarda el nombre del ciclo.
fecha_inicio_ciclo	DATE	Guarda la fecha de inicio de un ciclo.
fecha_fin_ciclo	DATE	Guarda la fecha de fin de un ciclo.
eliminado	TINYIN	Guarda un valor para conocer su estado: 1 eliminado, 2 no eliminado.

Tabla 32. Descripción de la tabla tb_ciclo.

Nombre: tb_ciclo_tb_area_salud		
Descripción: Guarda los datos esenciales de las áreas de salud actualizadas. Es el resultado de la unión entre las tablas tb_ciclo y tb_area_salud.		
Atributo	**Tipo**	**Descripción**

id_ciclo	INTEGER	Forma parte de la llave primaria. Importada de la tabla tb_ciclo.
id_area_salud	INTEGER	Forma parte de la llave primaria. Importada de la tabla tb_area_salud
id_unidad_salud	INTEGER	Forma parte de la llave primaria. Importada de la tabla tb_area_salud

Tabla 33. Descripción de la tabla tb_ciclo_tb_area_salud.

Nombre: tb_circunscripcion		
Descripción: Guarda los datos de las circunscripciones.		
Atributo	**Tipo**	**Descripción**
id_circunscripcion	INTEGER	Llave primaria de la tabla.
circunscripcion	VARCHAR	Guarda el nombre de la circunscripción.

Tabla 34. Descripción de la tabla tb_circunscripcion.

Nombre: tb_consejo_popular		
Descripción: Almacena los datos de los consejos populares.		
Atributo	**Tipo**	**Descripción**
id_consejo_popular	INTEGER	Llave primaria de la tabla.
consejo_popular	VARCHAR	Guarda el nombre del consejo popular.

Tabla 35. Descripción de la tabla tb_consejo_popular.

Nombre: tb_deposito		
Descripción: Guarda los datos de los depósitos.		
Atributo	**Tipo**	**Descripción**
id_deposito	INTEGER	Forma parte e la llave primaria de la tabla.
id_local_poitivo	INTEGER	Forma parte de la llave primaria. Importada de la tabla tb_local_positivo
id_inspeccion_diaria	INTEGER	Forma parte de la llave primaria. Importada de la tabla

		tb_inspeccion_diaria
fecha_inspecion	DATE	Forma parte de la llave primaria. Importada de la tabla tb_inspeccion_diaria
id_brigada	INTEGER	Forma parte de la llave primaria. Importada de la tabla n_brigada
id_area_salud	INTEGER	Forma parte de la llave primaria. Importada de la tabla tb_area_salud
id_unidad_salud	INTEGER	Forma parte de la llave primaria. Importada de la tabla tb_area_salud
id_nombre_deposito	INTEGER	Guarda el id del nombre del deposito.
lugar	VARCHAR	Guarda el lugar del deposito
cantidad_larvas_est_1	INTEGER	Guarda la cantidad de larvas encontradas en el estadío 1.
cantidad_larvas_est_2	INTEGER	Guarda la cantidad de larvas encontradas en el estadío 2.
cantidad_larvas_est_3	INTEGER	Guarda la cantidad de larvas encontradas en el estadío 3.
cantidad_larvas_est_4	INTEGER	Guarda la cantidad de larvas encontradas en el estadío 4.
cantidad_pupas	INTEGER	Guarda la cantidad de pupas encontradas.
cantidad_adultos	INTEGER	Guarda la cantidad de mosquitos adultos encontrados.

Tabla 36. Descripción de la tabla tb_deposito.

Nombre: tb_edificio		
Descripción: Guarda los datos de los edificios		
Atributo	**Tipo**	**Descripción**
id_local_positivo	INTEGER	Forma parte de la llave primaria. Importada de la tabla de la tabla tb_local_positivo

id_inspeccion_diaria	INTEGER	Forma parte de la llave primaria. Importada de la tabla tb_inspeccion_diaria
fecha_inspeccion	DATE	Forma parte de la llave primaria. Importada de la tabla tb_inspeccion_diaria
id_brigada	INTEGER	Forma parte de la llave primaria. Importada de la tabla tb_brigada
id_area_salud	INTEGER	Forma parte de la llave primaria. Importada de la tabla tb_area_salud
id_unidad_salud	INTEGER	Forma parte de la llave primaria. Importada de la tabla tb_area_salud
numero_edificio	INTEGER	Guarda el número del edificio.
numero_apartamento	VARCHAR	Guarda el número del apartamento.

Tabla 37. Descripción de la tabla tb_edificio.

Nombre: tb_especie_asociada		
Descripción: Guarda los datos de las especies asociadas a los locales positivos.		
Atributo	**Tipo**	**Descripción**
id_especie_asociada	INTEGER	Forma parte de la llave primaria de la tabla.
id_deposito	INTEGER	Forma parte de la llave primaria. Importada de la tabla tb_deposito
id_local_positivo	INTEGER	Forma parte de la llave primaria. Importada de la tabla tb_local_positivo
id_inspeccion_diaria	INTEGER	Forma parte de la llave primaria. Importada de la tabla tb_insp[eccion_diaria
fecha_inspeccion	DATE	Forma parte de la llave primaria. Importada de la tabla tb_insp[eccion_diaria
id_brigada	INTEGER	Forma parte de la llave primaria. Importada de la tabla n_brigada

id_area_salud	INTEGER	Forma parte de la llave primaria. Importada de la tabla
id_unidad_salud	INTEGER	Forma parte de la llave primaria. Importada de la tabla
id_especie	INTEGER	Forma parte de la llave primaria. Importada de la tabla
cantidad_larvas	INTEGER	Almacena la cantidad de larvas encontradas.
cantidad_pupas	INTEGER	Guarda la cantidad de pupas halladas.

Tabla 38. Descripción de la tabla tb_especie_asociada.

Nombre: tb_historico_area_ciclo		
Descripción: Guarda los datos esenciales de las áreas de salud desactualizadas		
Atributo	**Tipo**	**Descripción**
id_historico	INTEGER	Llave primaria de la tabla.
id_ciclo	INTEGER	Almacena el id del ciclo histórico, importado de la tabla tb_ciclo.
id_area_salud	INTEGER	Guarda el id del área de salud. Importado de la tabla tb_area_salud.
id_unidad_salud	INTEGER	Almacena el id de la unidad de salud. Importado de la tabla tb_area_salud.

Tabla 39. Descripción de la tabla tb_historico_area_ciclo.

Nombre: tb_inspeccion_diaria		
Descripción: Alamcena los datos de las inspecciones diarias		
Atributo	**Tipo**	**Descripción**
id_inspeccion_diaria	INTEGER	Forma parte de la llave primaria de la tabla.
fecha_inspeccion	DATE	Forma parte de la llave primaria. Importada de la tabla

		tb_inspeccion_diaria.
id_brigada	INTEGER	Forma parte de la llave primaria. Importada de la tabla n_brigada.
id_area_salud	INTEGER	Forma parte de la llave primaria. Importada de la tabla tb_area_salud.
id_unidad_salud	INTEGER	Forma parte de la llave primaria. Importada de la tabla tb_area_salud.
id_localidad	INTEGER	Forma parte de la llave primaria. Importada de la tabla tb_localidad.
id_autonumerico_manzana	INTEGER	Forma parte de la llave primaria. Importada de la tabla tb_manzana.
id_manzana	INTEGER	Forma parte de la llave primaria. Importada de la tabla tb_manzana.
cantidad_locales_inspeccionados_diario	INTEGER	Guarda la cantidad de locales inspeccionados en un día por una brigada.
cantidad_locales_inspeccionados_acumulado	INTEGER	Guarda la cantidad de locales inspeccionados hasta por una brigada hasta un día determinado dentro del ciclo.
cantidad_locales_cerrados_diario	INTEGER	Guarda la cantidad de locales que no pudieron ser inspeccionados por una brigada en un día.
cantidad_locales_cerrados_acumulado	INTEGER	Guarda la cantidad de locales que no han podido

		ser inspeccionados por determinada brigada hasta un día determinado dentro del ciclo.
cantidad_locales_recuperados_diario	INTEGER	Guarda la cantidad de locales que son recuperados por una brigada en un día.
cantidad_locales_recuperados_acumulado	INTEGER	Guarda la cantidad de locales que han sido recuperados por una brigada hasta un día determinado dentro del ciclo.
cantidad_depositos_inspeccionados_diario	INTEGER	Guarda la cantidad de depósitos que son inspeccionados por una brigada en un día.
cantidad_depositos_inspeccionados_acumulado	INTEGER	Guarda la cantidad de depósitos que han sido inspeccionados por una brigada hasta un día determinado dentro del ciclo.
cantidad_depositos_destruidos_diario	INTEGER	Guarda la cantidad de depósitos destruidos por una brigada en un día.
cantidad_depositos_destruidos_acumulado	INTEGER	Guarda la cantidad de depósitos que han sido destruidos por una brigada hasta un día determinado dentro del ciclo.

cantidad_depositos_tratados_diario	INTEGER	Guarda la cantidad de depósitos que son tratados por una brigada en un día.
cantidad_depositos_tratados_acumulado	INTEGER	Guarda la cantidad de depósitos que han sido tratados por una brigada hasta un día determinado dentro del ciclo.
cantidad_depositos_flameados_diario	INTEGER	Guarda la cantidad de depósitos que son flameados por una brigada en un día.
cantidad_depositos_flamedos_acumulado	INTEGER	Guarda la cantidad de depósitos que han sido flameados por una brigada hasta un día determinado dentro del ciclo.

Tabla 40. Descripción de la tabla tb_inspeccion_diaria.

Nombre: tb_local_positivo		
Descripción: Se guardan los datos de los locales encontrados positivos		
Atributo	**Tipo**	**Descripción**
id_local_positivo	INTEGER	Forma parte de la llave primaria de la tabla.
id_inspeccion_diaria	INTEGER	Forma parte de la llave primaria de la tabla. Es importada de tb_inspeccion_diaria.
fecha_insoeccion	DATE	Forma parte de la llave primaria de la tabla. Es importada de tb_inspeccion_diaria.
id_brigada	INTEGER	Forma parte de la llave primaria de a

		tabla. Importada de n_brigada.
id_area_salud	INTEGER	Forma parte de la llave primaria de a tabla. Importada de tb_area_salud.
id_unidad_salud	INTEGER	Forma parte de la llave primaria de a tabla. Importada de tb_area_salud.
fecha_diagnostico	DATE	Guarda la fecha de diagnóstico del local positivo.
tipo_local	VARCHAR	Especifica el tipo de local encontrado positivo (casa, edificio, terreno baldío)
id_consejo_popular	INTEGER	Guarda el id del consejo popular.
id_circunscripcion	INTEGER	Guarda el id de la circunscripción.
id_calle	INTEGER	Guarda el id de la calle de la dirección.
entre_calle_1	VARCHAR	Guarda el primer número de la entrecalle.
entre_calle_2	VARCHAR	Guarda el segundo número de la entrecalle.
id_etapa_colecta	INTEGER	Almacena el id de la etapa de colecta. Importado de la tabla n_etapa_colecta.
id_tipo_colector	INTEGER	Almacena el id del tipo de colector. Importado de la tabla n_tipo_colector.
zona_riesgo	TINYINT	Especifica si es una zona de riesgo o no.

Tabla 41. Descripción de la tabla tb_local_positivo.

Nombre: tb_localidad		
Descripción: Guarda los datos de las localidades		
Atributo	**Tipo**	**Descripción**
id_localidad	INTEGER	Llave primaria de la tabla.
localidad	VARCHAR	Guarda el nombre de la localidad.
id_municipio	INTEGER	Guarda el id del municipio. Importado de la tabla tb_municipio.
id_provincia	INTEGER	Guarda el id de la provincia. Importado

		de la tabla tb_municipio.

Tabla 42. Descripción de la tabla tb_localidad.

Nombre: tb_manzana		
Descripción: Guarda los datos de las manzanas de viviendas		
Atributo	**Tipo**	**Descripción**
id_autonumerico_manzana	INTEGER	Llave primaria de la taba.
id_manzana	INTEGER	Almacena el id de la manzana.
numero_manzana		Almacena el número de la manzana.
cantidad_locales_existentes	INTEGER	Guarda la cantidad de locales existentes en una manzana.
id_localidad	INTEGER	Almacena el id de la localidad.
eliminado	TINYINT	Guarda un valor que puede ser 1 (si el elemento está eliminado), 0 (si el elemento está activo)

Tabla 43. Descripción de la tabla tb_manzana.

Nombre: tb_municipio		
Descripción: Guarda los datos de los municipios		
Atributo	**Tipo**	**Descripción**
id_municipio	INTEGER	Llave primaria de la tabla.
id_provincia	INTEGER	Guarda el id de la provincia a la que el municipio pertenece.
municipio	VARCHAR	Guarda el nombre de l municipio.

Tabla 44. Descripción de la tabla n_municipio.

Nombre: tb_plan_larvitrampa		
Descripción: Almacena los datos de los planes de larvitrampas		
Atributo	**Tipo**	**Descripción**
id_plan_larvitrampa	INTEGER	Forma parte de la llave primaria de la tabla.
id_area_salud	INTEGER	Forma parte de la llave primaria

		de la tabla. Es importada de la tabla tb_area_salud.
id_unidad-salud	INTEGER	Forma parte de la llave primaria de la tabla. Es importada de tb_area_salud.
fecha_inicio	DATE	Almacena la fecha de inicio del plan de larvitrampas.
fecha_fin	DATE	Almacena la fecha de fin del plan de larvitrampas.
cantidad_larvitrampas_a_instalar	INTEGER	Guarda la cantidad de larvitrampas a instalar.
cantidad_larvitrampas_instaladas	INTEGER	Guarda la cantidad de larvitrampas, instaladas..
cantidad_larvitrampas_inspeccionadas	INTEGER	Guarda la cantidad de larvitrampas, inspeccionadas..
plan_completado	TINYINT	Especifica si el plan ha sido terminado o no.

Tabla 45. Descripción de la tabla tb_plan_larvitrampa.

Nombre: tb_provincia		
Descripción: Contiene los datos de las provincias.		
Atributo	**Tipo**	**Descripción**
id_provincia	INTEGER	Llave primaria de la tabla.
provincia	VARCHAR	Nombre de la provincia.

Tabla 45. Descripción de la tabla tb_provincia.

Nombre: tb_rol		
Descripción: Guarda los datos de los roles		
Atributo	**Tipo**	**Descripción**
id_rol	INTEGER	Llave primaria de la tabla.
desc_rol	VARCHAR	Breve descripción del rol.
desc_rol_full	CHAR	Descripción para mostrar al usuario.

eliminado	TINYINT	Se refiere a si fue eliminado o no.

Tabla 46. Descripción de la tabla tb_rol.

Nombre: tb_usuario_s3a		
Descripción: Guarda los usuarios del Sistema de Autenticación, Autorización y Auditoria		
Atributo	**Tipo**	**Descripción**
usuario	VARCHAR	Llave primaria de la tabla.

Tabla 47. Descripción de la tabla tb_usuario_s3a.

Nombre: tb_vinculo		
Descripción: Guarda todos los vínculos de la aplicación.		
Atributo	**Tipo**	**Descripción**
id_vinculo	INTEGER	Llave primaria de la tabla.
lesc_vinculo	VARCHAR	Breve descripción del vínculo.
link_ref	VARCHAR	Esta sería la dirección del vínculo referenciado.
activo	INTEGER	Se refiere a si está activo o no.

Tabla 48. Descripción de la tabla tb_vinculo.

Nombre: tb_vivienda		
Descripción: Guarda los datos de las viviendas.		
Atributo	**Tipo**	**Descripción**
Id_local_positivo	INTEGER	Forma parte de la llave primaria de la tabla. Importada de tb_local_positivo.
Id_inspeccion_diaria	INTEGER	Forma parte de la llave primaria de la tabla. Importada de tb_inspeccion_diaria.
Fecha_inspeccion	DATE	Almacena la fecha de la inspección.
Id_brigada	INTEGER	Almacena del id de la brigada que inspeccionó la vivienda. Importado de la tabla n_brigada.
Id_area_salud	INTEGER	Forma parte de la llave primaria de la

		tabla. Importada de tb_area_salud
Id_unidad_salud	INTEGER	Forma parte de la llave primaria de la tabla. Importada de tb_area_salud
Numero_vivienda	VARCHAR	Almacena el número de la vivienda.

Tabla 49. Descripción de la tabla tb_vivienda.

Nombre: usuario_s3a_rol		
Descripción: Guarda los roles de los usuarios del Sistema de Autenticación, Autorización y Auditoria		
Atributo	**Tipo**	**Descripción**
usuario	VARCHAR	Almacena el usuario.
id_rol	INTEGER	Guarda el rol que presenta el usuario.

Tabla 50. Descripción de la tabla usuario_s3a_rol.

3.5 Mecanismo de Seguridad

La seguridad informática consiste en asegurar que los recursos del sistema de información de una organización sean utilizados de la manera que se decidió y que el acceso a la información allí contenida así como su modificación sólo sea posible a las personas que se encuentren acreditadas y dentro de los límites de su autorización.

En la actualidad a la hora de desarrollar cualquier tipo de software informático el primer paso es asegurar la seguridad del mismo, con el objetivo de proteger la información que este manipule. Para el desarrollo del módulo Vectores del Sistema de Control Sanitario Internacional se tuvo en cuanta como objetivo fundamental el desarrollo de un mecanismo que permitiera asegurar que el acceso a la información solamente fuera realizado por los usuarios autorizados, para lo que se llevó a cabo un mecanismo de seguridad que se divide en dos partes: Autenticación y Autorización.

Aunque el componente SAAA del Registro Informatizado para la Salud se encarga de esto para la plataforma PLASER (Plataforma de Servicios) y se piensa extender a aplicaciones en otras plataformas este no satisface las necesidades de autorización que el sistema de Vectores necesita, debido a que en el SAAA un usuario tiene una lista de módulos a los que puede acceder y al ser el propio sistema de Vectores un módulo, este no gestiona la autorización dentro del mismo; debido a lo cual como parte de la estrategia de integración se hace uso de este componente para llevar a cabo la

Autenticación de los usuarios, mientras que la autorización se determina mediante un propio mecanismo implementado para el sistema de Control Sanitario Internacional basado en roles, donde se le asigna a un usuario uno o varios roles, los cuáles permiten a los usuarios tener determinados privilegios en la aplicación.

Estos privilegios se manejan a nivel de método o función, por lo que un rol tiene acceso a determinadas funciones en la aplicación, las cuales están contenidas en clases. Debido a esto un rol tiene acceso a una clase y a un método de la misma; lo cual está fundamentado por la forma en que se hacen las llamadas a los métodos en el framework CodeIgniter. En el mismo para tener acceso a un método solamente es necesario escribir en la barra de direcciones del navegador la dirección URL del sitio y seguido el nombre de la clase controladora y el método que se desea invocar.

Para determinar si un usuario posee o no permiso para acceder a determinado método se implementó en el constructor de la clase padre de todas las controladoras, MY_Controller, un mecanismo de verificación. Para determinar si los roles asignados al usuario, tienen permisos para invocar al método contenido en determinada clase. De ser cierto, la clase ejecuta el método solicitado, en caso contrario la misma redirecciona al usuario a la clase C_Usuarios; la cual posee un método denominado No_Acces que se encarga de comunicarle el error de autorización al usuario en cuestión.

Conclusiones

En el capítulo se han expuesto las estrategias a seguir para la integración con otros componentes existentes en el SISalud. Fueron mostradas las descripciones de las clases del sistema para lograr un mejor entendimiento de las mismas, y la descripción de las tablas de la Base de Datos, de la cual también se presentaron los modelos lógicos y físicos. Además, se explicó el mecanismo de seguridad implementado en el sistema.

Conclusiones Generales

Con la culminación del presente trabajo de diploma se dio cumplimiento a los objetivos propuestos. Obteniéndose como resultados: la implementación del módulo Vectores y la obtención de una interfaz gráfica que cumple con los estándares predefinidos para los software de la salud, satisfaciendo las necesidades del cliente.

Para ello fue necesario hacer un análisis de las herramientas a utilizar y adiestrarse en el uso de las mismas, aprovechando sus bondades.

Además fue de gran importancia la obtención de los modelos de implementación, despliegue y base de datos; sin los cuales, la implementación y acoplamiento de las clases del sistema y el trabajo con la base de datos hubiese sido pobre o imposible.

Fue muy importante también la integración con el SISalud, sistema que comprende módulos que son de vital importancia para la existencia del sistema en cuestión, como son el Sistema de Autenticación, Autorización y Auditoría (SAAA), el Registro de Ubicación (RU), el Registro de Localidades (RL) y el Registro de Unidades de Salud (RUS).-

Recomendaciones

Los autores recomiendan:

- Valorar el uso de Yahoo User Interface (YUI), herramienta potente basada en la técnica de programación AJAX.
- Desarrollar otras funcionalidades correspondientes al trabajo de la Campaña Antivectorial y el control de los recursos materiales utilizados en dicha campaña.
- Para próximas versiones integrar al sistema funcionalidades relacionadas al proceso de gestión de la información relacionada con el control no sólo del mosquito *Aedes Aegypti,* sino también de otros vectores como los roedores, las moscas y cucarachas.
- Incluir trabajo con cartografía para la localización de regiones positivas al mosquito *Aedes Aegypti*.
- Mantener sobre el sistema el proceso de mantenimiento y actualización adecuado, logrando de esta manera una mayor fiabilidad del sistema y de la información que con el mismo se gestiona.

Referencias bibliográficas

1. Dimagin Web Development. *Dimagin.* [En línea] [Citado el: 15 de enero de 2008.] http://www.dimagin.net/es/contenido.php?t_id=6.

2. blogold. [En línea] [Citado el: 15 de enero de 2008.] http://www.avidos.net/blogold/aplicaciones-web/.

3. **Vegas, Jesús.** El Navegador Web, Browser . *Departamento de Informática Universidad Valladolid.* [En línea] [Citado el: 11 de diciembre de 2007.] http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node19.html.

4. *Ídem a Referencia 3.*

5. *Ídem a Referencia 3.*

6. **Guglielmetti, Marcos.** *Master Magazine.* [En línea] [Citado el: 14 de diciembre de 2007.] http://www.mastermagazine.info/termino/5288.php.

7. *Ídem a Referencia 6.*

8. **Vegas, Jesús.** HyperText Transfer Protocol, HTTP . *Departamento de Informática Universidad de Valladolid.* [En línea] [Citado el: 6 de diciembre de 2007.] http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node13.html.

9. —. El Servidor Web . *Departamento de Informática Universidad de Valladolid.* [En línea] [Citado el: 15 de enero de 2008.] http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node20.html.

10. **masadelante.com.** ¿Qué es un servidor web (Web Servers)? *masadelante.com.* [En línea] [Citado el: 15 de enero de 2008.] http://www.masadelante.com/faq-servidor-web.htm.

11. **Ciberaula.** Una Introducción a APACHE. *Ciberaula.* [En línea] [Citado el: 6 de febrero de 2008.] http://linux.ciberaula.com/articulo/linux_apache_intro.

12. *Ídem a Referencia 11.*

13. *Ídem a Referencia 11.*

14. **Alvarez, Miguel Angel.** Qué es Javascript. *desarrolloweb.com.* [En línea] [Citado el: 8 de febrero de 2008.] http://www.desarrolloweb.com/articulos/25.php.

15. *Ídem a Referencia 14.*

16. *Ídem a Referencia 14.*

17. **Torre, Aníbal de la.** Lenguajes del lado servidor o cliente. [En línea] 2006. [Citado el: 8 de febrero de 2008.] http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html.

Referencias Bibliográficas

18. **Váldes, Damián Pérez.** Los diferentes lenguajes de programación para la web. *Maestros del Web.* [En línea] 2 de noviembre de 2007. [Citado el: 20 de febrero de 2008.] http://www.maestrosdelweb.com/principiantes/los-diferentes-lenguajes-de-programacion-para-la-web/.

19. *Ídem a Referencia 18.*

20. *Ídem a Referencia 18.*

21. *Ídem a Referencia 18.*

22. *Ídem a Referencia 17.*

23. **Proaño, Diego Javier Burbano.** *MySQL hispano.* [En línea] [Citado el: 26 de febrero de 2008.] http://www.mysql-hispano.org/articulos/num43/analisis-comparativo.pdf.

24. **Martínez, Daniel Pecos.** PostGreSQL vs. MySQL. *NET Pecos.* [En línea] [Citado el: 3 de marzo de 2008.] http://www.netpecos.org/docs/mysql_postgres/x15.html#AEN30.

25. **Eguiluz, Javier.** El framework Symfony, una introducción práctica . *Maestros del Web.* [En línea] 6 de septiembre de 2007. [Citado el: 11 de marzo de 2008.] http://www.maestrosdelweb.com/editorial/el-framework-symfony-una-introduccion-practica-i-parte/.

26. **Gutiérrez, Javier J.** *Departamento de Lenguajes y Sistemas Informáticos Universidad de Sevilla.* [En línea] [Citado el: 28 de marzo de 2008.] http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

27. *Ídem a Referencia 26.*

28. **ExpressionEngine.** CodeIgniter User Guide. *CodeIgniter.* [En línea] [Citado el: 2 de abril de 2008.] http://codeigniter.com/user_guide/.

29. *Ídem a Referencia 28.*

30. *Ídem a Referencia 28.*

31. *Ídem a Referencia 28.*

32. *Ídem a Referencia 25.*

33. *Ídem a Referencia 25.*

34. *Ídem a Referencia 25.*

35. *Ídem a Referencia 25.*

36. **YAHOO .** The Yahoo! User Interface Library (YUI). *YAHOO DEVELOPER NETWORK.* [En línea] [Citado el: 22 de abril de 2008.] http://developer.yahoo.com/yui/.

37. **Maldonado, Daniel M.** El CoDiGoK: Que son los IDE de Programación . *El CoDiGoK.* [En línea] 3 de septiembre de 2007. [Citado el: 6 de mayo de 2008.] http://elcodigok.blogspot.com/2007/09/que-son-los-ide-de-programacin.html.

38. *Ídem a Referencia 37.*

39. *Ídem a Referencia 37.*

40. *Ídem a Referencia 37.*

41. *Ídem a Referencia 37.*

42. **Alvarez, Miguel Angel.** Zend Studio. *desarrolloweb.com.* [En línea] [Citado el: 13 de mayo de 2008.] http://www.desarrolloweb.com/articulos/1178.php.

43. —. Evaluando Zend Studio. *Maestros del Web.* [En línea] [Citado el: 22 de abril de 2008.] http://www.maestrosdelweb.com/editorial/zendstudio/.

44. *Notepad++.* [En línea] [Citado el: 14 de mayo de 2008.] http://notepad-plus.sourceforge.net/es/site.htm.

45. **Reynoso, Carlos Billy.** *Introducción a la arquitectura de software.* 2004.

46. **Terreros, Julio Casal.** Desarrollo de Software basado en Componentes. [En línea] [Citado el: 30 de mayo de 2008.] http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/MTJ_3985/default.aspx.

47. *Ídem a Referencia 46.*

48. **Mayoral, Antonio Gutiérrez.** Patrón de diseño MVC. *Departamento de Sistemas Telemáticos y Computación.* [En línea] 11 de febrero de 2005. [Citado el: 26 de mayo de 2008.] http://gsyc.es/~agutierr/pfc-tecnica-html/node43.html.

Bibliografía**.**

- —. El Servidor Web . *Departamento de Informática Universidad de Valladolid.* [En línea] [Citado el: 15 de enero de 2008.] http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node20.html.
- **2006.** Programación Web. *www.lenguajes-de-programacion.com.* [Online] 2006. [Cited: febrero 5, 2008.] http://www.lenguajes-de-programacion.com/programacion-web.shtml.
- **Alvarez, Miguel Angel.** Qué es ASP. *www.desarrolloweb.com.* [Online] [Cited: febrero 15, 2008.] http://www.desarrolloweb.com/articulos/393.php .
- blogold. [En línea] [Citado el: 15 de enero de 2008.] http://www.avidos.net/blogold/aplicaciones-web/.
- **Burbano, Diego Javier. 2006.** *www.mysql-hispano.org.* [Online] mayo 17, 2006. [Cited: febrero 22, 2008.] http://www.mysql-hispano.org/page.php?id=43.
- **Ciberaula.** Una Introducción a APACHE. *Ciberaula.* [En línea] [Citado el: 6 de febrero de 2008.] http://linux.ciberaula.com/articulo/linux_apache_intro.
- Conceptos básicos del servidor web. *www.cibernetia.com.* [Online] [Cited: enero 15, 2008.] http://www.cibernetia.com/manuales/instalacion_servidor_web/1_conceptos_basicos.php.
- Dimagin Web Development. *Dimagin.* [En línea] [Citado el: 15 de enero de 2008.] http://www.dimagin.net/es/contenido.php?t_id=6.
- **Guglielmetti, Marcos.** *Master Magazine.* [En línea] [Citado el: 14 de diciembre de 2007.] http://www.mastermagazine.info/termino/5288.php.
- HISTORIA DE INTERNET. *homepage.mac.com.* [Online] [Cited: enero 18, 2008.] http://homepage.mac.com/xe1ac/albanet/articulos/HISTORIA.htm.
- *homepage.mac.com.* [Online] [Cited: marzo 3, 2008.] http://homepage.mac.com/imaz/iblog/C612772037/E20050907222635/Media/Algunos%20Tipos%20de%20Arquitecturas.pdf.
- **Ivar Jacobson, Grady Booch and James Rumbaugh. 2000.** *El Proceso Unificado de Desarrollo de Software.* Madrid : PEARSON EDUCACION, 2000.
- **Larman, Craig.** Introducción al análisis y diseño orientado a objetos. Ciudad Habana : Félix Varela, 2004.
- **Lenn, Bass, C., P. and Kazman, Rick.** *Software Architecture in Practice.* s.l. : Addison-Wesley Professional, 2003.
- **Lidia Fuentes, Troya, José M and Vallecillo, Antonio.** *www.lcc.uma.es.* [Online] [Cited: febrero 5, 2008.] http://www.lcc.uma.es/~av/Docencia/Doctorado/tema1.pdf.

Bibliografía

- **masadelante.com.** ¿Qué es un servidor web (Web Servers)? *masadelante.com.* [En línea] [Citado el: 15 de enero de 2008.] http://www.masadelante.com/faq-servidor-web.htm.
- **Pecos, Daniel.** PostGreSQL vs. MySQL. *www.netpecos.org.* [Online] [Cited: febrero 22, 2008.] http://www.netpecos.org/docs/mysql_postgres/x15.html#AEN30.
- **Pérez, Carlos Torres.** Vista de Implementación. *ewh.ieee.org.* [Online] [Cited: marzo 10, 2008.] http://ewh.ieee.org/r9/guadalajara/boletin/marzo02/vistaimpl.htm.
- **Pressman, Roger.** *Ingeniería del Software. Un enfoque práctico.* Ciudad Habana : Félix Varela, 2005.
- **Ricardo, Febe Ángel Ciudad.** Utilización del patrón modelo – vista – controlador (mvc) en el diseño de software educativos. [En línea] 2007. http://www.informaticahabana.com/evento_virtual/?q=node/223&ev=3er%20Congreso%20Internacional%20de%20Tecnolog%C3%ADas,%20Contenidos%20Multimedia.
- **Rodríguez, César Colado. 2003.** *www.germinus.com.* [Online] febrero 2003. [Cited: diciembre 15, 2008.] http://www.germinus.com/sala_prensa/articulos/Diseno_desarr_aplicaciones_web_multidispo%20(Febrero%202003).pdf.
- **Torre, Aníbal de la. 2006.** Lenguajes del lado servidor o cliente. *www.adelat.com.* [Online] 2006. [Cited: febrero 5, 2008.] http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html.
- **Váldes, Damián Pérez.** Los diferentes lenguajes de programación para la web. *Maestros del Web.* [En línea] 2 de noviembre de 2007. [Citado el: 20 de febrero de 2008.] http://www.maestrosdelweb.com/principiantes/los-diferentes-lenguajes-de-programacion-para-la-web/.
- **Vázquez, José Antonio Gallego**. Desarrollo Web con PHP y MySQL. Madrid : Ediciones Anaya Multimedia, 2003.
- **Vegas, Jesús.** El Navegador Web, Browser . *Departamento de Informática Universidad Valladolid.* [En línea] [Citado el: 11 de diciembre de 2007.] http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node19.html.
- **Vegas, Jesús.** HyperText Transfer Protocol, HTTP . *Departamento de Informática Universidad de Valladolid.* [En línea] [Citado el: 6 de diciembre de 2007.] http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node13.html.
- *WWW.JTECH.UA.ES.* [ONLINE] HTTP://WWW.JTECH.UA.ES/J2EE/EJEMPLOS/JSP/SESION01-APUNTES.HTM.

Glosario de Términos

Diagrama Entidad Relación: Es un modelo que representa a la realidad a través de un esquema gráfico empleando los terminología de entidades, que son objetos que existen y son los elementos principales que se identifican en el problema a resolver con el diagramado y se distinguen de otros por sus características particulares denominadas atributos, el enlace que rige la unión de las entidades está representada por la relación del modelo.

Enfermedades exóticas: Dígase de las enfermedades procedentes de otros países que pueden ser introducidas de manera intencional o no, a un hábitat fuera de su hábitat normal.

Hospederos: Personas que portan el germen de una enfermedad y son transmisoras de ella.

Inspecciones diarias: Mecanismo mediante el cual se identifica los focos de *Aedes Aegypti* y con ellos los locales positivos para su posterior tratamiento.

Locales positivos: Es un lugar o sitio, sea local de tipo casa, edificio, centro de trabajo o terreno baldío, que presenta focos de mosquitos *Aedes Aegypti*.

MINSAP: Ministerio de Salud Publica, institución de nuestro país perteneciente a la rama de la salud.

Navegador: Los buscadores exploradores o navegadores son programas, generalmente gratuitos, que instalados en el ordenador permiten ver documentos almacenados en el disco duro, disquete, etc., o a través de Internet, acceder a documentos alojados en servidores web.

Plan Larvitrampa: Ambiente o hábitat artificial construido con la intención de probar o comprobar la existencia de mosquitos *Aedes Aegypti* principalmente. Estas larvitrampas se construyen con neumáticos en desuso, botellas u otros recipientes propensos a convertirse en ambiente favorable para la proliferación de este tipo de vector. Siempre deberán marcarse y revisarse para evitas así que se convierta en un foco.

Protocolos denominado TCP/IP: Son una colección completa de protocolos de comunicación de datos que ha conseguido la hegemonía en las redes actuales gracias a su utilización en Internet. Son protocolos abiertos, disponibles públicamente y desarrollados con independencia del hardware y del

sistema operativo. Proporcionan un esquema de direccionamiento que permite identificar de manera unívoca a cualquier sistema dentro de una red.

Servidor Web: Básicamente, un servidor web sirve contenido estático a un navegador, carga un archivo y lo sirve a través de la red al navegador de un usuario. Este intercambio es mediado por el navegador y el servidor que hablan el uno con el otro mediante HTTP.

Vectores: Especie portadora o huésped intermedio de un parásito o virus que transmite el germen de una enfermedad a otro huésped. Los vectores por excelencia son las cucarachas, ratones, mosquitos y moscas.

WEB (WWW): Red de documentos HTML intercomunicados y distribuidos entre servidores del mundo entero.